Morphological Inflection Generation Using Character Sequence to Sequence Learning

Manaal Faruqui[1]   **Yulia Tsvetkov**[1]   **Graham Neubig**[2]   **Chris Dyer**[1]

[1]Language Technologies Institute, Carnegie Mellon University, USA

[2]Graduate School of Information Science, Nara Institute of Science and Technology, Japan

{mfaruqui,ytsvetko,cdyer}@cs.cmu.edu neubig@is.naist.jp

Abstract

Morphological inflection generation is the task of generating the inflected form of a given lemma corresponding to a particular linguistic transformation. We model the problem of inflection generation as a character sequence to sequence learning problem and present a variant of the neural encoder-decoder model for solving it. Our model is language independent and can be trained in both supervised and semi-supervised settings. We evaluate our system on seven datasets of morphologically rich languages and achieve either better or comparable results to existing state-of-the-art models of inflection generation.

1 Introduction

Inflection is the word-formation mechanism to express different grammatical categories such as tense, mood, voice, aspect, person, gender, number and case. Inflectional morphology is often realized by the concatenation of bound morphemes (prefixes and suffixes) to a root form or stem, but nonconcatenative processes such as ablaut and infixation are found in many languages as well. Table 1 shows the possible inflected forms of the German stem *Kalb* (calf) when it is used in different cases and numbers. The inflected forms are the result of both ablaut (e.g., $a \rightarrow \ddot{a}$) and suffixation (e.g., +*ern*).

Inflection generation is useful for reducing data sparsity in morphologically complex languages. For example, statistical machine translation suffers from data sparsity when translating morphologically-rich languages, since every surface form is considered an

	singular	plural
nominative	Kalb	Kälber
accusative	Kalb	Kälber
dative	Kalb	Kälbern
genitive	Kalbes	Kälber

Table 1: An example of an inflection table from the German noun dataset for the word *Kalb* (calf).

independent entity. Translating into lemmas in the target language, and then applying inflection generation as a post-processing step, has been shown to alleviate the sparsity problem (Minkov et al., 2007; Toutanova et al., 2008; Clifton and Sarkar, 2011; Fraser et al., 2012; Chahuneau et al., 2013a). Modeling inflection generation has also been used to improve language modeling (Chahuneau et al., 2013b), identification of multi-word expressions (Oflazer et al., 2004), among other applications.

The traditional approach to modeling inflection relies on hand-crafted finite state transducers and lexicography, e.g., using *two-level morphology* (Koskenniemi, 1983; Kaplan and Kay, 1994). Such systems are appealing since they correspond to linguistic theories, but they are expensive to create, they can be fragile (Oflazer, 1996), and the composed transducers can be impractically large. As an alternative, machine learning models have been proposed to generate inflections from root forms as string transduction (Yarowsky and Wicentowski, 2000; Wicentowski, 2004; Dreyer and Eisner, 2011; Durrett and DeNero, 2013; Ahlberg et al., 2014; Hulden, 2014; Ahlberg et al., 2015; Nicolai et al., 2015). However, these impose either assumptions about the set of possible morphological processes
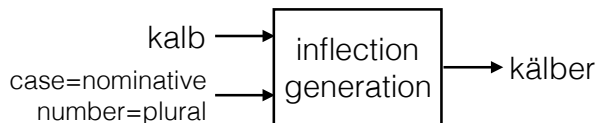
Figure 1: A general inflection generation model.

(e.g. affixation) or require careful feature engineering.

In this paper, we present a model of inflection generation based on a neural network sequence to sequence transducer. The root form is represented as sequence of characters, and this is the input to an encoder-decoder architecture (Cho et al., 2014; Sutskever et al., 2014). The model transforms its input to a sequence of output characters representing the inflected form (§4). Our model makes no assumptions about morphological processes, and our features are simply the individual characters. The model is trained on pairs of root form and inflected forms obtained from inflection tables extracted from Wiktionary.[1] We improve the supervised model with unlabeled data, by integrating a character language model trained on the vocabulary of the language.

Our experiments show that the model achieves better or comparable results to state-of-the-art methods on the benchmark inflection generation tasks (§5). For example, our model is able to learn long-range relations between character sequences in the string aiding the inflection generation process required by Finnish vowel harmony (§6), which helps it obtain the current best results in that language.

2   Inflection Generation: Background

Durrett and DeNero (2013) formulate the task of supervised inflection generation for a given root form, based on a large number of training inflection tables extracted from Wiktionary. Every inflection table contains the inflected form of a given root word corresponding to different linguistic transformations (cf. Table 1). Figure 1 shows the inflection generation framework. Since the release of the Wiktionary dataset, several different models have reported performance on this dataset. As we are also using this dataset, we will now review these models.

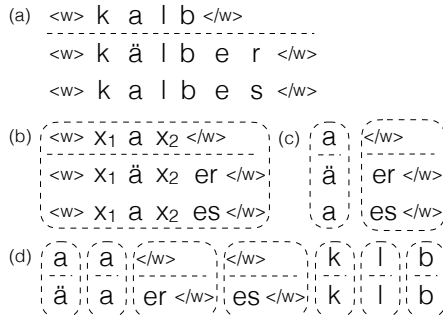We denote the models of Durrett and DeNero

Figure 2: Rule extraction: (a) Character aligned-table; (b) Table-level rule of AFH14, AFH15 (c) Vertical rules of DDN13 and (d) Atomic rules of NCK15.

(2013), Ahlberg et al. (2014), Ahlberg et al. (2015), and Nicolai et al. (2015), by DDN13, AFH14, AFH15, and NCK15 respectively. These models perform inflection generation as string transduction and largely consist of three major components: (1) Character alignment of word forms in a table; (2) Extraction of string transformation rules; (3) Application of rules to new root forms.

The first step is learning character alignments across inflected forms in a table. Figure 2 (a) shows alignment between three word forms of *Kalb*. Different models use different heuristic algorithms for alignments such as edit distance, dynamic edit distance (Eisner, 2002; Oncina and Sebban, 2006), and longest subsequence alignment (Bergroth et al., 2000). Aligning characters across word forms provide spans of characters that have changed and spans that remain unchanged. These spans are used to extract rules for inflection generation for different inflection types as shown in Figure 2 (b)–(d).

By applying the extracted rules to new root forms, inflected words can be generated. DDN13 use a semi-Markov model (Sarawagi and Cohen, 2004) to predict what rules should be applied, using character n-grams ($n = 1$ to 4) as features. AFH14 and AFH15 use substring features extracted from words to match an input word to a rule table. NCK15 use a semi-Markov model inspired by DDN13, but additionally use target n-grams and joint n-grams as features sequences while selecting the rules.

Motivation for our model. Morphology often makes references to segmental features, like place or manner of articulation, or voicing status (Chom-

sky and Halle, 1968). While these can be encoded as features in existing work, our approach treats segments as vectors of features "natively". Our approach represents every character as a bundle of continuous features, instead of using discrete surface character sequence features. Also, our model uses features as part of the transduction rules themselves, whereas in existing work features are only used to rescore rule applications.

In existing work, the learner implicitly specifies the class of rules that can be learned, such as "delete" or "concatenate". To deal with phenomenona like segment lengthening in English: *run* → *running*; or reduplication in Hebrew: *Kelev* → *Klavlav*, *Chatul* → *Chataltul*; (or consonant gradation in Finnish), where the affixes are induced from characters of the root form, one must engineer a new rule class, which leads to poorer estimates due to data sparsity. By modeling inflection generation as a task of generating a character sequence, one character at a time, we do away with such problems.

3 Neural Encoder-Decoder Models

Here, we describe briefly the underlying framework of our inflection generation model, called the recurrent neural network (RNN) encoder-decoder (Cho et al., 2014; Sutskever et al., 2014) which is used to transform an input sequence $\vec{x}$ to output sequence $\vec{y}$. We represent an item by x, a sequence of items by $\vec{x}$, vectors by $\mathbf{x}$, matrices by $\mathbf{X}$, and sequences of vectors by $\vec{\mathbf{x}}$.

3.1 Formulation

In the encoder-decoder framework, an encoder reads a variable length input sequence, a sequence of vectors $\vec{\mathbf{x}} = \langle \mathbf{x}_1, \cdots, \mathbf{x}_T \rangle$ (corresponding to a sequence of input symbols $\vec{x} = \langle x_1, \cdots, x_T \rangle$) and generates a fixed-dimensional vector representation of the sequence. $\mathbf{x}_t \in \mathbb{R}^l$ is an input vector of length l. The most common approach is to use an RNN such that:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) \quad (1)$$

where $\mathbf{h}_t \in \mathbb{R}^n$ is a hidden state at time t, and f is generally a non-linear transformation, producing $\mathbf{e} := \mathbf{h}_{T+1}$ as the input representation. The decoder is trained to predict the next output y_t given the

encoded input vector $\mathbf{e}$ and all the previously predicted outputs $\langle y_1, \cdots y_{t-1} \rangle$. In other words, the decoder defines a probability over the output sequence $\vec{y} = \langle y_1, \cdots, y_{T'} \rangle$ by decomposing the joint probability into ordered conditionals:

$$p(\vec{y}|\vec{x}) = \prod_{t=1}^{T'} p(y_t|\mathbf{e}, \langle y_1, \cdots, y_{t-1} \rangle) \quad (2)$$

With a decoder RNN, we can first obtain the hidden layer at time t as: $\mathbf{s}_t = g(\mathbf{s}_{t-1}, \{\mathbf{e}, \mathbf{y}_{t-1}\})$ and feed this into a softmax layer to obtain the conditional probability as:

$$p(y_t = i|\mathbf{e}, \vec{y}_{<t}) = \text{softmax}(\mathbf{W}_s \mathbf{s}_t + \mathbf{b}_s)_i \quad (3)$$

where, $\vec{y}_{<t} = \langle y_1, \cdots, y_{t-1} \rangle$. In recent work, both f and g are generally LSTMs, a kind of RNN which we describe next.

3.2 Long Short-Term Memory (LSTM)

In principle, RNNs allow retaining information from time steps in the distant past, but the nonlinear "squashing" functions applied in the calculation of each $\mathbf{h}_t$ result in a decay of the error signal used in training with backpropagation. LSTMs are a variant of RNNs designed to cope with this "vanishing gradient" problem using an extra memory "cell" (Hochreiter and Schmidhuber, 1997; Graves, 2013). Past work explains the computation within an LSTM through the metaphors of deciding how much of the current input to pass into memory or forget. We refer interested readers to the original papers for details.

4 Inflection Generation Model

We frame the problem of inflection generation as a sequence to sequence learning problem of character sequences. The standard encoder-decoder models were designed for machine translation where the objective is to translate a sentence (sequence of words) from one language to a semantically equivalent sentence (sequence of words) in another language. We can easily port the encoder-decoder translation model for inflection generation. Our model predicts the sequence of characters in the inflected string given the characters in the root word (input).

However, our problem differs from the above setting in two ways: (1) the input and output character
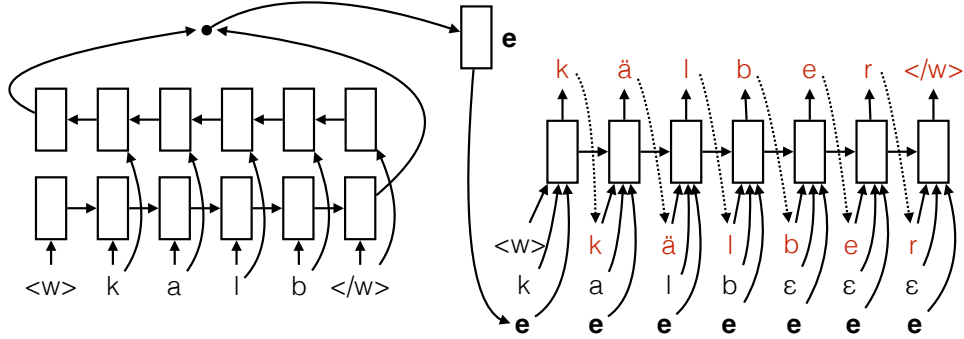
Figure 3: The modified encoder-decoder architecture for inflection generation. Input characters are shown in black and predicted characters are shown in red. • indicates the append operation.

sequences are mostly similar except for the inflections; (2) the input and output character sequences have different semantics. Regarding the first difference, taking the word *play* as an example, the inflected forms corresponding to past tense and continuous forms are *played* and *playing*. To better use this correspondence between the input and output sequence, we also feed the input sequence directly into the decoder:

$$\mathbf{s}_t = g(\mathbf{s}_{t-1}, \{\mathbf{e}, \mathbf{y}_{t-1}, \mathbf{x}_t\}) \qquad (4)$$

where, g is the decoder LSTM, and $\mathbf{x}_t$ and $\mathbf{y}_t$ are the input and output character vectors respectively. Because the lengths of the input and output sequences are not equal, we feed an ϵ character in the decoder, indicating null input, once the input sequence runs out of characters. These ϵ character vectors are parameters that are learned by our model, exactly as other character vectors.

Regarding the second difference, to provide the model the ability to learn the transformation of semantics from input to output, we apply an affine transformation on the encoded vector $\mathbf{e}$:

$$\mathbf{e} \leftarrow \mathbf{W}_{trans}\mathbf{e} + \mathbf{b}_{trans} \qquad (5)$$

where, $\mathbf{W}_{trans}, \mathbf{b}_{trans}$ are the transformation parameters. Also, in the encoder we use a bi-directional LSTM (Graves et al., 2005) instead of a uni-directional LSTM, as it has been shown to capture the sequence information more effectively (Ling et al., 2015; Ballesteros et al., 2015; Bahdanau et al., 2015). Our resultant inflection generation model is shown in Figure 3.

4.1 Supervised Learning

The parameters of our model are the set of character vectors, the transformation parameters ($\mathbf{W}_{trans}, \mathbf{b}_{trans}$), and the parameters of the encoder and decoder LSTMs (§3.2). We use negative log-likelihood of the output character sequence as the loss function:

$$-\log p(\vec{y}|\vec{x}) = -\sum_{t=1}^{T'} \log p(y_t|\mathbf{e}, \vec{y}_{<t}) \qquad (6)$$

We minimize the loss using stochastic updates with AdaDelta (Zeiler, 2012). This is our purely supervised model for inflection generation and we evaluate it in two different settings as established by previous work:

Factored Model. In the first setting, we learn a separate model for each type of inflection independent of the other possible inflections. For example, in case of German nouns, we learn 8, and for German verbs, we learn 27 individual encoder-decoder inflection models (cf. Table 3). There is no parameter sharing across these models. We call these factored models of inflection generation.

Joint Model. In the second setting, while learning a model for an inflection type, we also use the information of how the lemma inflects across all other inflection types i.e., the inflection table of a root form is used to learn different inflection models. We model this, by having the same encoder in the encoder-decoder model across all inflection models.[2] The encoder in our model is learning a

---

[2]We also tried having the same encoder and decoder across inflection types, with just the transformation matrix being dif-

$p_{\text{LM}}(\vec{y})$	$p(\vec{y}|\vec{x})$
len($\vec{y}$) - len($\vec{x}$)	levenshtein($\vec{y}, \vec{x}$)
same-suffix($\vec{y}, \vec{x}$)?	subsequence($\vec{y}, \vec{x}$)?
same-prefix($\vec{y}, \vec{x}$)?	subsequence($\vec{x}, \vec{y}$)?

Table 2: Features used to rerank the inflected outputs. $\vec{x}$, $\vec{y}$ denote the root and inflected character sequences resp.

Dataset	root forms	Infl.
German Nouns (DE-N)	2764	8
German Verbs (DE-V)	2027	27
Spanish Verbs (ES-V)	4055	57
Finnish NN & Adj. (FI-NA)	6400	28
Finnish Verbs (FI-V)	7249	53
Dutch Verbs (NL-V)	11200	9
French Verbs (FR-V)	6957	48

Table 3: The number of root forms and types of inflections across datasets.

representation of the input character sequence. Because all inflection models take the same input but produce different outputs, we hypothesize that having the same encoder can lead to better estimates.

4.2 Semi-supervised Learning

The model we described so far relies entirely on the availability of pairs of root form and inflected word form for learning to generate inflections. Although such supervised models can be used to obtain inflection generation models (Durrett and DeNero, 2013; Ahlberg et al., 2015), it has been shown that unlabeled data can generally improve the performance of such systems (Ahlberg et al., 2014; Nicolai et al., 2015). The vocabulary of the words of a language encode information about what correct sequences of characters in a language look like. Thus, we learn a language model over the character sequences in a vocabulary extracted from a large unlabeled corpus. We use this language model to make predictions about the next character in the sequence given the previous characters, in following two settings.

Output Reranking. In the first setting, we first train the inflection generation model using the supervised setting as described in §4.1. While making predictions for inflections, we use beam search to generate possible output character sequences and rerank them using the language model probability along with other easily extractable features as described in Table 2. We use pairwise ranking optimization (PRO) to learn the reranking model (Hopkins and May, 2011). The reranker is trained on the beam output of dev set and evaluated on test set.

Language Model Interpolation. In the second setting, we interpolate the probability of observing the next character according to the language model with the probability according to our inflection gen-

---

ferent (equ. 5), and observed consistently worse results.

eration model. Thus, the loss function becomes:

$$-\log p(\vec{y}|\vec{x}) = \frac{1}{Z} \sum_{t=1}^{T'} - \log p(y_t|\mathbf{e}, \vec{y}_{<t})$$
$$- \lambda \log p_{\text{LM}}(y_t|\vec{y}_{<t}) \quad (7)$$

where $p_{LM}(y_t|\vec{y}_{<t})$ is the probability of observing the word y_t given the history estimated according to a language model, $\lambda \geq 0$ is the interpolation parameter which is learned during training and Z is the normalization factor. This formulation lets us use any off-the-shelf pre-trained character language model easily (details in §5).

4.3 Ensembling

Our loss functions (equ. 6 & 7) formulated using a neural network architecture are non-convex in nature and are thus difficult to optimize. It has been shown that taking an ensemble of models which were initialized differently and trained independently leads to improved performance (Hansen and Salamon, 1990; Collobert et al., 2011). Thus, for each model type used in this work, we report results obtained using an ensemble of models. So, while decoding we compute the probability of emitting a character as the product-of-experts of the individual models in the ensemble: $p_{ens}(y_t|\cdot) = \frac{1}{Z} \prod_{i=1}^{k} p_i(y_t|\cdot)^{\frac{1}{k}}$ where, $p_i(y_t|\cdot)$ is the probability according to i-th model and Z is the normalization factor.

5 Experiments

We now conduct experiments using the described models. Note that not all previously published models present results on all settings, and thus we compare our results to them wherever appropriate.

Hyperparameters. Across all models described in this paper, we use the following hyperparameters.

In both the encoder and decoder models we use single layer LSTMs with the hidden vector of length 100. The length of character vectors is the size of character vocabulary according to each dataset. The parameters are regularized with ℓ_2, with the regularization constant 10^{-5}.[3] The number of models for ensembling are $k = 5$. Models are trained for at most 30 epochs and the model with best result on development set is selected.

5.1 Data

Durrett and DeNero (2013) published the Wiktionary inflection dataset with training, development and test splits. The development and test sets contain 200 inflection tables each and the training sets consist of the remaining data. This dataset contains inflections for German, Finnish and Spanish. This dataset was further augmented by (Nicolai et al., 2015), by adding Dutch verbs extracted from CELEX lexical database (Baayen et al., 1995), French verbs from Verbsite, an online French conjugation dictionary and Czech nouns and verbs from the Prague Dependnecy Treebank (Hajič et al., 2001). As the dataset for Czech contains many incomplete tables, we do not use it for our experiments. These datasets come with pre-specified training/dev/test splits, which we use. For each of these sets, the training data is restricted to 80% of the total inflection tables, with 10% for development and 10% for testing. We list the size of these datasets in Table 3.

For semi-supervised experiments, we train a 5-gram character language model with Witten-Bell smoothing (Bell et al., 1990) using the SRILM toolkit (Stolcke, 2002). We train the character language models on the list of unique word types extracted from the Wikipedia dump for each language after filtering out words with characters unseen in the inflection generation training dataset. We obtained around 2 million unique words for each language.

5.2 Results

Supervised Models. The individual inflected form accuracy for the factored model (§4.1) is shown in Table 4. Across datasets, we obtain either com-

	DDN13	NCK15	Ours
DE-V	94.76	**97.50**	96.72
DE-N	88.31	**88.60**	88.12
ES-V	99.61	99.80	**99.81**
FI-V	97.23	**98.10**	97.81
FI-NA	92.14	93.00	**95.44**
NL-V	90.50	96.10	**96.71**
FR-V	98.80	**99.20**	98.82
Avg.	94.47	96.04	**96.20**

Table 4: Individual form prediction accuracy for **factored supervised** models.

	DDN13	AFH14	AFH15	Ours
DE-V	96.19	97.01	**98.11**	97.25
DE-N	88.94	87.81	**89.88**	88.37
ES-V	99.67	99.52	**99.92**	99.86
FI-V	96.43	96.36	97.14	**97.97**
FI-NA	93.41	91.91	93.68	**94.71**
Avg.	94.93	94.53	**95.74**	95.63
NL-V	93.88	–	–	96.16
FR-V	98.60	–	–	98.74
Avg.	95.30	–	–	**96.15**

Table 5: Individual form prediction accuracy for **joint supervised** models.

parable or better results than NCK15 while obtaining on average an accuracy of 96.20% which is higher than both DDN13 and NCK15. Our factored model performs better than DDN13 and NCK15 on datasets with large training set (ES-V, FI-V, FI-NA, NL-V, FR-V) as opposed to datasets with small training set (DE-N, DE-V). In the joint model setting (cf. Table 5), on average, we perform better than DDN13 and AFH14 but are behind AFH15 by 0.11%. Our model improves in performance over our factored model for DE-N, DE-V, and ES-V, which are the three smallest training datasets. Thus, parameter sharing across different inflection types helps the low-resourced scenarios.[4]

Semi-supervised Models. We now evaluate the utility of character language models in inflection generation, in two different settings as described earlier (§4.2). We use the factored model as our base model in the following experiments as it performed

---

[3]Using dropout did not improve our results.

[4]Although NCK15 provide results in the joint model setting, they also use raw data in the joint model which makes it incomparable to our model and other previous models.

|	AFH14	NCK15	Interpol	Rerank
DE-V	97.87	**97.90**	96.79	97.11
DE-N	91.81	**89.90**	88.31	89.31
ES-V	99.58	99.90	99.78	**99.94**
FI-V	96.63	**98.10**	96.66	97.62
FI-NA	93.82	93.60	94.60	**95.66**
Avg.	**95.93**	95.88	95.42	**95.93**
NL-V	–	96.60	**96.66**	96.64
FR-V	–	**99.20**	98.81	98.94
Avg.	–	**96.45**	96.08	**96.45**

Table 6: Individual form prediction accuracy for **factored semi-supervised** models.

Model	Accuracy
Encoder-Decoder	79.08
Encoder-Decoder Attention	95.64
Ours W/O Encoder	84.04
Ours	**96.20**

Table 7: Avg. accuracy across datasets of the encoder-decoder, attentional encoder-decoder & our model without encoder.

better than the joint model (cf. Table 4 & 5). Our reranking model which uses the character language model along with other features (cf. Table 2) to select the best answer from a beam of predictions, improves over almost all the datasets with respect to the supervised model and is equal on average to AFH14 and NCK15 semi-supervised models with 96.45% accuracy. We obtain the best reported results on ES-V and FI-NA datasets (99.94% and 95.66% respectively). However, our second semi-supervised model, the interpolation model, on average obtains 96.08% and is surprisingly worse than our supervised model (96.20%).

Comparison to Other Architectures. Finally it is of interest how our proposed model compares to more traditional neural models. We compare our model against a standard encoder-decoder model, and an encoder-decoder model with attention, both trained on root form to inflected form character sequences. In a standard encoder-decoder model (Sutskever et al., 2014), the encoded input sequence vector is fed into the hidden layer of the decoder as input, and is not available at every time step in contrast to our model, where we additionally feed in $\mathbf{x}_t$ at every time step as in equ. 4. An attentional model computes a weighted average of the hidden layer of
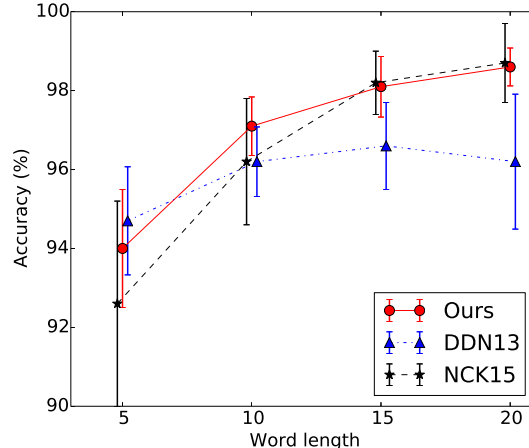
Figure 4: Plot of inflection prediction accuracy against the length of gold inflected forms. The points are shown with minor offset along the x-axis to enhance clarity.

the input sequence, which is then used along with the decoder hidden layer to make a prediction (Bahdanau et al., 2015). These models also do not take the root form character sequence as inputs to the decoder. We also evaluate the utility of having an encoder which computes a representation of the input character sequence in a vector $\mathbf{e}$ by removing the encoder from our model in Figure 3. The results in Table 7 show that we outperform the encoder-decoder model, and the model without an encoder substantially. Our model is slightly better than the attentional encoder-decoder model, and is simpler as it does not have the additional attention layer.

6 Analysis

Length of Inflected Forms. In Figure 4 we show how the prediction accuracy of an inflected form varies with respect to the length of the correct inflected form. To get stable estimates, we bin the inflected forms according to their length: < 5, $[5, 10)$, $[10, 15)$, and ≥ 15. The accuracy for each bin is macro-averaged across 6 datasets[5] for our factored model and the best models of DDN13 and NCK15. Our model consistently shows improvement in performance as word length increases and is significantly better than DDN13 on words of length more than 20 and is approximately equal to NCK15. On words of length < 5, we perform worse than DDN13

---

[5]We remove DE-N as its the smallest and shows high variance in results.

but better than NCK15. On average, our model has the least error margin across bins of different word length as compared to both DDN13 and NCK15. Using LSTMs in our model helps us make better predictions for long sequences, since they have the ability to capture long-range dependencies.

Finnish Vowel Harmony. Our model obtains the current best result on the Finnish noun and adjective dataset, this dataset has the longest inflected words, some of which are > 30 characters long. Finnish exhibits *vowel harmony*, i.e, the occurrence of a vowel is controlled by other vowels in the word. Finnish vowels are divided into three groups: front (ä, ö, y), back (a, o, u), and neutral (e, i). If back vowels are present in a stem, then the harmony is back (i.e, front vowels will be absent), else the harmony is front (i.e, back vowels will be absent). In compound words the suffix harmony is determined by the final stem in the compound. For example, our model correctly inflects the word *fasisti* (fascist) to obtain *fasisteissa* and the compound *tärkkelyspitoinen* (starch containing) to *tärkkelyspitoisissa*. The ability of our model to learn such relations between these vowels helps capture vowel harmony. For FI-NA, our model obtains 99.87% for correctly predicting vowel harmony, and NCK15 obtains 98.50%.We plot the character vectors of these Finnish vowels (cf. Figure 5) using t-SNE projection (van der Maaten and Hinton, 2008) and observe that the vowels are correctly grouped with visible transition from the back to the front vowels.

7 Related Work

Similar to the encoder in our framework, Rastogi et al. (2016) extract sub-word features using a forward-backward LSTM from a word, and use them in a traditional weighted FST to generate inflected forms. Neural encoder-decoder models of string transduction have also been used for sub-word level transformations like grapheme-to-phoneme conversion (Yao and Zweig, 2015; Rao et al., 2015).

Generation of inflectional morphology has been particularly useful in statistical machine translation, both in translation from morphologically rich languages (Goldwater and McClosky, 2005), and into morphologically rich languages (Minkov et al., 2007; Toutanova et al., 2008; Clifton and Sarkar,
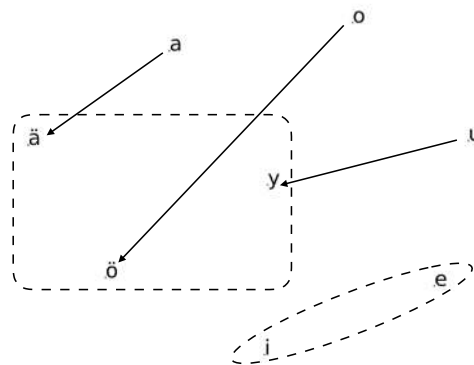
Figure 5: Plot of character vectors of Finnish vowels. Their organization shows that front, back and neutral vowel groups have been discovered. The arrows show back and front vowel correspondences.

2011; Fraser et al., 2012). Modeling the morphological structure of a word has also shown to improve the quality of word clusters (Clark, 2003) and word vector representations (Cotterell and Schütze, 2015).

Inflection generation is complementary to the task of morphological and phonological segmentation, where the existing word form needs to be segmented to obtained meaningful sub-word units (Creutz and Lagus, 2005; Snyder and Barzilay, 2008; Poon et al., 2009; Narasimhan et al., 2015; Cotterell et al., 2015; Cotterell et al., 2016). An additional line of work that benefits from implicit modeling of morphology is neural character-based natural language processing, e.g., part-of-speech tagging (Santos and Zadrozny, 2014; Ling et al., 2015) and dependency parsing (Ballesteros et al., 2015). These models have been successful when applied to morphologically rich languages, as they are able to capture word formation patterns.

8 Conclusion

We have presented a model that generates inflected forms of a given root form using a neural network sequence to sequence string transducer. Our model obtains state-of-the-art results and performs at par or better than existing inflection generation models on seven different datasets. Our model is able to learn long-range dependencies within character sequences for inflection generation which makes it specially suitable for morphologically rich languages.

Acknowledgements

References

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *Proc. of EACL*.

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. *Proc. of NAACL*.

Harald R. Baayen, Richard Piepenbrock, and Leon Gulikers. 1995. *The CELEX Lexical Database. Release 2 (CD-ROM)*. LDC, University of Pennsylvania.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proc. of EMNLP*.

Timothy C Bell, John G Cleary, and Ian H Witten. 1990. *Text compression*. Prentice-Hall, Inc.

Lasse Bergroth, Harri Hakonen, and Timo Raita. 2000. A survey of longest common subsequence algorithms. In *Proc. of SPIRE*.

Victor Chahuneau, Eva Schlinger, Noah A. Smith, and Chris Dyer. 2013a. Translating into morphologically rich languages with synthetic phrases. In *Proc. of EMNLP*.

Victor Chahuneau, Noah A Smith, and Chris Dyer. 2013b. Knowledge-rich morphological priors for bayesian language models. In *Proc. of NAACL*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proc. of EMNLP*.

N. Chomsky and M. Halle. 1968. *The Sound Pattern of English*. Harper & Row, New York, NY.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proc. of EACL*.

Ann Clifton and Anoop Sarkar. 2011. Combining morpheme-based machine translation with post-processing morpheme prediction. In *Proc. of ACL*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proc. of NAACL*.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2015. Modeling word forms using latent underlying morphs and phonology. *Transactions of the Association for Computational Linguistics*, 3:433–447.

Ryan Cotterell, Tim Vieria, and Hinrich Schütze. 2016. A joint model of orthography and morphological segmentation. In *Proc. of NAACL*.

Mathias Creutz and Krista Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a dirichlet process mixture model. In *Proc. of EMNLP*.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proc. of NAACL*.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proc. of ACL*.

Alexander Fraser, Marion Weller, Aoife Cahill, and Fabienne Cap. 2012. Modeling inflection and word-formation in SMT. In *Proc. of EACL*.

Sharon Goldwater and David McClosky. 2005. Improving statistical MT through morphological analysis. In *Proc. of EMNLP*, pages 676–683.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *Proc. of ICANN*.

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

Jan Hajič, Barbora Vidová-Hladká, and Petr Pajas. 2001. The Prague Dependency Treebank: Annotation structure and support. In *Proc. of the IRCS Workshop on Linguistic Databases*.

Lars Kai Hansen and Peter Salamon. 1990. Neural network ensembles. In *Proc. of PAMI*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proc. of EMNLP*.

Mans Hulden. 2014. Generalizing inflection tables into paradigms with finite state operations. In *Proc. of the Joint Meeting of SIGMORPHON and SIGFSM*.

Ronald M Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational linguistics*, 20(3):331–378.

Kimmo Koskenniemi. 1983. Two-level morphology: A general computational model for word-form recognition and production. *University of Helsinki*.

Wang Ling, Tiago Luís, Luís Marujo, Rámon Fernandez 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 *Proc. of EMNLP*.

Einat Minkov, Kristina Toutanova, and Hisami Suzuki. 2007. Generating complex morphology for machine translation. In *Proc. of ACL*.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2015. An unsupervised method for uncovering morphological chains. *TACL*.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *Proc. of NAACL*.

Kemal Oflazer, Özlem çetinoğlu, and Bilge Say. 2004. Integrating morphology with multi-word expression processing in turkish. In *Proc. of the Workshop on Multiword Expressions*.

Kemal Oflazer. 1996. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1):73–89.

Jose Oncina and Marc Sebban. 2006. Learning stochastic edit distance: Application in handwritten character recognition. *Pattern recognition*, 39(9):1575–1587.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proc. of NAACL*.

Kanishka Rao, Fuchun Peng, Hasim Sak, and Françoise Beaufays. 2015. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *Proc. of ICASSP*.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proc. of NAACL*.

Cicero D. Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proc. of ICML*.

Sunita Sarawagi and William W Cohen. 2004. Semi-markov conditional random fields for information extraction. In *Proc. of NIPS*.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *In The Annual Conference of the*.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Proc. of Interspeech*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS*.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. 2008. Applying morphology generation models to machine translation. In *Proc. of ACL*, pages 514–522.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Richard Wicentowski. 2004. Multilingual noise-robust supervised morphological analysis using the word-frame model. In *Proc. of SIGPHON*.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. In *Proc. of ICASSP*.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proc. of ACL*.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.