Character-based recurrent neural networks for morphological relational reasoning

Olof Mogren¹* and Richard Johansson²
¹ olof@mogren.one, RISE Research institutes of Sweden
² richard.johansson@gu.se, University of Gothenburg, Sweden

ABSTRACT

We present a model for predicting inflected word forms based on morphological analogies. Previous work includes rule-based algorithms that determine and copy affixes from one word to another, with limited support for varying inflectional patterns. In related tasks such as morphological reinflection, the algorithm is provided with an explicit enumeration of morphological features which may not be available in all cases. In contrast, our model is feature-free: instead of explicitly representing morphological features, the model is given a demo pair that implicitly specifies a morphological relation (such as write:writes specifying infinitive:present). Given this demo relation and a query word (e.g. watch), the model predicts the target word (e.g. watches). To address this task, we devise a character-based recurrent neural network architecture using three separate encoders and one decoder.

Our experimental evaluation on five different languages shows that the exact form can be predicted with high accuracy, consistently beating the baseline methods. Particularly, for English the prediction accuracy is 94.85%. The solution is not limited to copying affixes from the demo relation, but generalizes to words with varying inflectional patterns, and can abstract away from the orthographic level to the level of morphological forms.

* A large portion of this work was done while O.M. was a PhD student at Chalmers university of technology, Sweden.
Analogical reasoning is an important part of human cognition (Gentner et al. 2001). Resolving analogies by mapping unknown data points to known analogous examples allows us to draw conclusions about the previously unseen data points. This is closely related to zero-shot and one-shot learning: strategies that are useful when training data is very limited, such as when explicit labels may not be available, or only sparsely available. In linguistics, analogies have been studied extensively, e.g. phonetic analogies (Yvon 1997) and semantic analogies (Mikolov et al. 2013a). In general, an analogy is defined as a quadruple of objects $A$, $B$, $C$, and $D$ having the analogical relation: $A$ is to $B$ as $C$ is to $D$, and the problem is to predict $D$ given $A$, $B$, and $C$. In this work, we study morphological analogies where $A$, $B$, $C$, and $D$ are words. The pair $(A, B)$ represents a demo relation representing some morphological transformation between two word forms, and the problem is to transform the query word $C$ from the source form to the target form as specified by the demo relation. The task may be illustrated with a simple example: see is to sees as eat is to what?

A good solver for morphological analogies can be of practical help as writing aids for authors, suggesting synonyms in a form specified by examples rather than using explicitly specified forms. Furthermore, models that can generate words with correct inflection are important building blocks for many tasks within natural language processing. Studying how systems can learn the right abstraction to generate inflected words using limited supervision, can help us learn how to create systems for more complex language generation tasks, such as machine translation, automatic summarization, and dialogue systems.

Previous work has tackled the problem of predicting the target form by identifying the string transformation (insertions, deletions, or replacements of characters) in the demo relation, and then trying to apply the same transformation to $C$ (Lepage 1998). For instance, this algorithm correctly solves the example given above, since it just needs to add an $s$ to the query word.

However, such solutions are brittle, as they are unable to abstract away from the raw strings, failing when the given relation is realized differently in $A$ and $B$ than in $C$ and $D$. On a basic level, the model needs to take into account phonological processes such as umlaut and
vowel harmony, as well as orthographic quirks such as the rule in English that turns y into ie in certain contexts. Furthermore, an even more challenging problem is that the model will need to take into account that words belong to groups whose inflectional patterns are different – morphological paradigms. In all these cases, to be successful, a solution needs to abstract away from the raw character-based representation to a higher level representation of the relations.

In this work, we propose a supervised machine learning approach to the problem of predicting the target word in a morphological analogy. The model is based on character-level recurrent neural networks (RNNs), which have recently seen much success in a number of morphological prediction tasks (Faruqui et al. 2016; Kann and Schütze 2016). This model is able to go beyond the simple string substitutions handled by previous approaches: it picks up contextual string transformations including orthographic and phonological rules, and is able to generalize between inflection paradigms.

Machine learning approaches, including character-based RNNs, have been successfully applied in several types of prediction problems in morphology, including lemmatization, inflection and reinflection (see Section 2.2). However, those tasks have either been more restricted than ours (e.g. lemmatization), or relied on an explicit enumeration of morphological features, which may not be available in all cases. In contrast, our model is a completely feature-free approach to generating inflected forms, which can predict any form in a morphological paradigm.

The fact that our model does not rely on explicit features makes it applicable in scenarios with under-resourced languages where such annotations may not be available. However, since the model is trained using a weaker signal than in the traditional feature-based scenario, it needs to learn a latent representation from the analogies that play the same role as the morphological features otherwise would, making the task more challenging.

2 RELATED WORK

Analogical reasoning is useful in many different tasks. In this section we will limit the survey to work that is relevant to morphological applications.
2.1 Morphological analogies

Lepage (1998) presented an algorithm to solve morphological analogies by analyzing three input words, determining changes in prefixes, infixes, and suffixes, and adding or removing them to or from the query word, transforming it into the target:

$$\text{reader:unreadable} = \text{doer}:x \rightarrow x = \text{undoable}.$$ 

Stroppa and Yvon (2005) presented algebraic definitions of analogies and a solution for analogical learning as a two-step process: learning a mapping from a memorized situation to a new situation, and transferring knowledge from the known to the unknown situation. The solution takes inspiration from k-nearest neighbour (k-NN) search, where, given a query $q$, one looks for analogous objects $A, B, C$ from the training data, and selects a suitable output based on a mapping of $A, B, C$ from input space to output space. The task studied in these papers is the same as in the current paper. The solutions, are however much limited in the generality. Our solution can learn very flexible relations and different inflectional patterns.

2.2 Character based modeling for morphology

The 2016 and 2017 SIGMORPHON shared tasks on morphological inflection (Cotterell et al. 2016a, 2017) have spurred some recent interest in morphological analysis. In this task, a word is given in one form, and should be transformed into a form specified by an explicit feature representation. These features represent number, gender, case, tense, aspect, etc. In comparison, the problem of morphological analogies is more difficult, as no explicit tags are provided: the forms must instead be inferred from a demo relation.

While morphological inflection tasks have previously been studied using rule-based systems (Koskenniemi 1984; Ritchie et al. 1991) and learned string transducers (Yarowsky and Wicentowski 2000; Nicolai et al. 2015a; Ahlberg et al. 2015; Durrett and DeNero 2013), they have more recently been dominated by character-level neural network models (Faruqui et al. 2016; Kann and Schütze 2016) as they address the inherent drawbacks of traditional models that represent words as atomic symbols. This offers a number of advantages: the vocabulary in a character-based model can be much smaller, as it only
The Neural morphological analogy system

needs to represent a finite and fairly small alphabet, and as long as
the characters are in the alphabet, no words will be out-of-vocabulary
(OOV). Character-level models can capture distributional properties,
not only of frequent words but also of words that occur rarely (Luong
and Manning 2016), and they need no tokenization, freeing the sys-
tem from one source of errors. Neural models working on character-
or subword-level have been applied in several natural language pro-
cessing (NLP) tasks, ranging from relatively basic tasks such as text
categorization (Zhang et al. 2015) and language modelling (Kim et al.
2016) to complex prediction tasks, such as translation (Luong and
Manning 2016; Sennrich et al. 2016). Because they can recognize pat-
terns on a subword level, character-based neural models are attractive
in NLP tasks that require an awareness of morphology.

2.3 Other morphological transformations

Lemmatization is the task of predicting the base form (lemma) of an
inflected word. A lemmatizer may make use of the context to get
(implicit) information about the source form of the word (Kosken-
niemi 1984; Kanis and Müller 2005; Chrupala et al. 2008; Jongejan
and Dalianis 2009; Chakrabarty et al. 2017). In comparison, our task
does not offer contextual information, but instead provides the (sim-
ilarly implicit) cues for the forms from the demo relation. With this
in mind, predicting the lemma is just a special case of the morpho-
logical analogy problem. Paradigm filling is the more general task of
predicting all unknown forms in a paradigm (Dreyer and Eisner 2011).

2.4 Morphological relations in word embedding models

Word analogies have been proposed as a way to demonstrate the util-
ity of neural word embeddings and to evaluate their quality (Mikolov
et al. 2013a; Mnih and Kavukcuoglu 2013; Nicolai et al. 2015b; Pen-
nington et al. 2014). Such embeddings show simple linear relation-
ships in the resulting continuous embedding space that allow for
finding impressive analogous relations such as

\[ v(\text{king}) - v(\text{man}) + v(\text{woman}) \approx v(\text{queen}) \]

where \( v(w) \) is the word embedding of the word \( w \). Analogies have been
categorized as either semantic or syntactic. (The example with “king”
and “queen” is a semantic analogy, while syntactic analogies relate
different morphological forms of the same words). Google’s dataset for syntactic analogies (Mikolov et al. 2013a) was proposed as a task to evaluate word embedding models on English.

Cotterell et al. (2016b) presented an approach using a Gaussian graphical model to process word embeddings computed using a standard toolkit such as Word2Vec to improve the quality of embeddings for infrequent words, and to construct embeddings for morphological forms that were missing in the training data (but belonging to a paradigm that had some form or forms in the data).

3 THE NEURAL MORPHOLOGICAL ANALOGY SYSTEM

In this paper, we present the Neural morphological analogy system (NMAS), a neural approach for morphological relational reasoning. We use a deep recurrent neural network with gated recurrent unit (GRU) cells that take words represented by their raw character sequences as input.

3.1 Morphological relational reasoning with analogies

We define the task as follows. Given a query word $q$ and a demo word in two forms $w_1$ and $w_2$, demonstrating a transformation from one word form to another, and where $q$ is another word in the same form as $w_1$, the task is to transform $q$ into the form represented by $w_2$.

3.2 Recurrent neural networks

A recurrent neural network (RNN) is an artificial neural network that can model a sequence of arbitrary length. Gated RNNs were proposed to solve some issues of basic “vanilla” RNNs (the difficulty to capture long dependencies and vanishing gradients) (Hochreiter 1998; Bengio et al. 1994). The long short term memory (LSTM) (Schmidhuber and Hochreiter 1997) is one of the most famous types. At every step in the sequence, it has a cell with three learnable gates that controls what parts of the internal memory vector to keep (the forget gate), what parts of the input vector to store in the internal memory (the input gate), and what to include in the output vector (the output gate). The gated recurrent unit (GRU) (Cho et al. 2014a) is a simplification of this approach, having only two gates by replacing the input and
The Neural morphological analogy system

forget gates with an update gate that simply erases memory whenever it is updating the state with new input. Hence, the GRU has fewer parameters, and still obtains similar performance as the original LSTM.

An RNN can easily be trained to predict the next token in a sequence, and when applied to words this essentially becomes a language model. A sequence-to-sequence model is a neural language model conditioned on another input sequence. Such a model can be trained to translate from one sequence to another (Sutskever et al. 2014; Cho et al. 2014b). This is the major building block in modern neural machine translation systems, where they are combined with an attention mechanism to help with the alignment (Bahdanau et al. 2015).

In language settings it is common to have a linear input layer that learns embeddings for a vocabulary of words. However, these models suffer from the limitations of having fixed word vocabularies, and being unable to learn subword patterns. As an alternative, an RNN can work either using a vocabulary of subword units, or a vocabulary of characters, as is the case in this paper.

3.3 Model layout

The proposed model has three major parts, the relation encoder, the query encoder, and the decoder, all working together to generate the predicted target form given the three input words: the demo relation \((w_1, w_2)\), and the query word \(q\). The whole model is trained end-to-end and requires no other input than the raw character sequences of the three input words \(w_1, w_2,\) and \(q\).

A. The relation encoder. The first part encodes the demo relation \(R_{\text{demo}} = (w_1, w_2)\) using an encoder RNN for each of the two words \(w_1\) and \(w_2\). The relation encoder RNNs share weights but have separate internal state representations. The outputs of the relation encoders are fed into a fully connected layer with tanh activation \(FC\) relation:

\[
h_{\text{rel}} = \tanh(W_{\text{rel}}[g_{\text{rel}}(0, w_1), g_{\text{rel}}(0, w_2)]),
\]

where \(g_{\text{rel}}\) is the output from the relation encoder RNN (using zero vectors as initial hidden states), \(w_1, w_2\) are sequences of one-hot encodings for the characters of \(w_1\) and \(w_2\), \(W_{\text{rel}}\) is the weight matrix for
the FC relation layer, and tanh is the element-wise nonlinearity. Here, \([x, y]\) means the concatenation of the vectors \(x\) and \(y\).

B. The query encoder. The query word \(q\) is encoded separately using a distinct encoder RNN. The final output from the query encoder is fed together with the output from FC relation (A) through a second fully connected layer (with tanh activation) FC combined:

\[
h_{\text{comb}} = \tanh(W_{\text{comb}}[h_{\text{rel}}, g_q(0, q)]),
\]

where \(h_{\text{rel}}\) is the output from FC relation, \(g_q\) is the output from the query RNN encoder, \(q\) is a sequence of embeddings of the characters of the query word, \(W_{\text{comb}}\) is the weight matrix for the FC combined layer, and tanh is the element-wise nonlinearity. The result \(h_{\text{comb}}\) is fed as the initial hidden state into the RNN decoder.

C. The decoder. The decoder RNN employs a standard attention mechanism (Bahdanau et al. 2015), computing a weighted sum of the sequence of outputs of the query encoder at every step \(t_d\) in the generation process. For each step \(t_e\) in the query encoder, the attention weight is computed using a multi-layer perceptron taking the decoder state at \(t_d\) and the query encoder state at \(t_e\) as inputs. For each decoder step \(t_d\), the output character is decided by computing a distribution over the alphabet using the softmax output layer, and then sampling greedily from this distribution; this is fast and has yielded good results. The distribution \(p(y_{t_d} = i) = h_{\text{dec};t_d}^{(i)}\) for each character \(i\) in the alphabet and for each step \(t_d\) in the decoder is modelled using:

\[
h_{\text{dec};t_d} = s(W_{\text{dec}}[g_{\text{dec}}(h_{\text{comb}}, y(0:t_d-1)), a]),
\]

where \(h_{\text{comb}}\) is the output from FC combined (used as the initial hidden state for the decoder RNN), \(g_{\text{dec}}\) is the output from the decoder RNN, \(y(0:t_d-1)\) is a sequence of embeddings of the characters generated by the decoder until step \(t_d-1\), \(W_{\text{dec}}\) is the weight matrix for the decoder output layer, \(a\) is the weighted sum of hidden states from the query encoder RNN computed by the attention mechanism, and \(s\) is the softmax activation function: \(s(z) = \frac{e^z}{\sum_i e^z_i}\). The result \(h_{\text{dec};t_d}\) is a vector that sums to one, defining the distribution over the alphabet at time \(t_d\).

[100]
The whole model is similar to a sequence-to-sequence model used for translation, with the addition of the relation encoder. Figure 1 shows the architecture of the model pictorially.

4 EXPERIMENTAL SETUP

This section explains the setup of the empirical evaluation of our model: how it is designed, trained, and evaluated.

The model was implemented using Pytorch;¹ all source code is freely available.² With the final hyperparameter settings (see Section 4), the model contains approximately 155,000 parameters, and it can be trained in a few hours on a modern GPU. In the experiments reported in this paper, the code was executed on a desktop PC with an NVIDIA Titan X GPU using Ubuntu 18.04.

The hyperparameters relevant to the proposed model are presented in Table 1. The RNN hidden size parameter decides the dimensionality of all four RNNs in the model, as we noticed no performance gain from varying them individually.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Explored</th>
<th>Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding size</td>
<td>50–350</td>
<td>100</td>
</tr>
<tr>
<td>FC relation size</td>
<td>50–350</td>
<td>100</td>
</tr>
<tr>
<td>FC combined size</td>
<td>50–350</td>
<td>200</td>
</tr>
<tr>
<td>RNN hidden size</td>
<td>25–350</td>
<td>100</td>
</tr>
<tr>
<td>RNN depth</td>
<td>1–3</td>
<td>2</td>
</tr>
<tr>
<td>Learning rate</td>
<td>1 × 10⁻³</td>
<td></td>
</tr>
<tr>
<td>L2 weight decay</td>
<td>5 × 10⁻⁵</td>
<td></td>
</tr>
<tr>
<td>Drop probability</td>
<td>0.0, 0.2, ..., 0.8</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 1: Hyperparameters in the model

Training was done with backpropagation through time (BPTT) and minibatch learning with the Adam optimizer (Kingma and Ba 2015). For each example in a minibatch, a relation type is selected uniformly randomly. Then two word pairs are selected randomly from that relation type; one of these will be the demo relation, and one will be the query–target pair. The output from the decoder (see

¹http://pytorch.org/
²https://github.com/olofmogren/char-rnn-wordrelations
Figure 1: The layout of the proposed model. The demo relation is encoded using two encoder RNNs with shared weights for the two demo word forms. A fully connected (FC) layer FC relation follows the demo relation pair. The query word is encoded separately, and its embedding is concatenated with the output from the fully connected (FC) layer FC relation, and fed as the initial hidden state into the RNN decoder which generates the output while using an attention pointer to the query encoder. In the example see is to sees as eat is to what, see is fed into the demo relation RNN encoder 1, sees is fed into the demo relation RNN encoder 2, eat is fed into the query RNN encoder, and the whole model is trained to generate the correct output eats at the decoder RNN.
The Neural morphological analogy system

Section 3.3 C) is a categorical distribution over the alphabet. We use the cross-entropy loss function for each character at position \( t_d \) in the output word as the learning objective for all parameters \( \theta \) in the model:

\[
\mathcal{L}(\theta) = -\frac{1}{N} \sum_{t_d} \mathbf{y}_{(t_d)} \cdot \log h_{\theta; t_d},
\]

where \( N \) is the length of the target word, \( \mathbf{y}_{(t_d)} \) is the one-hot encoding of the true target at position \( c \) and \( h_{\theta; t_d} \) is the model output distribution at position \( c \). Training duration was decided using early stopping (Wang et al. 1994).

One model with separate parameters was trained per language. The parameters are shared between the two encoding RNNs in the relation encoder, but the query encoder RNN and the decoder RNN have separate weights, as the model search showed best performance using this configuration. Ensembling did not improve the results.

4.1 Baselines

The task considered in this work is closely related to morphological reinflection. Systems trained for the latter task generally obtain higher absolute numbers of prediction accuracy than ours, because more information is given through the explicit enumeration of morphological tags. Our task is also related to the syntactic analogy task used to evaluate word embeddings (Mikolov et al. 2013c), and we also include the word embedding-based word-level analogy solution as a baseline.

Lepage. This baseline was implemented from the description in (Lepage 1998). The algorithm is rule-based, and uses information collected when computing edit distance between \( w_1 \) and \( w_2 \), as well as between \( w_1 \) and \( q \) (Wagner and Fischer 1974). It can handle changes in prefix, infix, and suffix, but fails when words exhibit different inflectional patterns.

Word embedding baseline. This baseline uses pre-trained word embeddings using Word2Vec CBOW (continuous bag-of-words) (Mikolov et al. 2013b) and FastText (Bojanowski et al. 2017), referred to as W2V and FT, respectively. The FastText embeddings are designed to take subword information into account, and they performed better
than Word2Vec CBOW-based vectors in our experiments. The prediction is selected by choosing the word in the vocabulary that has an embedding with the highest cosine similarity compared to

\[ \mathbf{v}(q) = \mathbf{v}(w_1) + \mathbf{v}(w_2), \]

where \( \mathbf{v}(w) \) is the word embedding of the word \( w \), \( q \) is the query word, and \( w_1, w_2 \) are the two demo words in the demo relation.

Word embeddings have been used in previous work for this task (then called syntactic analogies), but the solution is limited by a fixed vocabulary, and needs retraining to incorporate new words. Although it is trained without supervision, training requires much data and comparing the resulting vector above with all words in the vocabulary is expensive.

In this work, pretrained embeddings were downloaded and used. The utilized Word2Vec CBOW embeddings were downloaded from Kyubyong Park’s repository. The embeddings were trained using data from Wikipedia. The FastText embeddings used were downloaded from the FastText authors’ website. The embeddings were trained using data from CommonCrawl and Wikipedia. All the embeddings used have 300 dimensions.

To make the word embedding baseline stronger, we used the LePage baseline as a fallback whenever any of the three input words are missing in the vocabulary.

4.2 Datasets

One model was trained and evaluated on each of five different languages. Data for all languages except for English and Swedish was taken from the SIGMORPHON 2016 dataset (Cotterell et al. 2016a). The code for downloading the data and performing dataset split is available for download.

English. A total of 10 relations and their corresponding inverse relations were considered:

- nouns:
  - singular–plural, e.g. dog–dogs

---

3 https://github.com/Kyubyong/wordvectors/
4 https://fasttext.cc/
5 https://github.com/olofmogren/char-rnn-wordrelations/
The Neural morphological analogy system

- adjectives:
  - positive–comparative, e.g. high–higher
  - positive–superlative, e.g. high–highest
  - comparative–superlative, e.g. higher–highest

- verbs:
  - infinitive–past, e.g. sit–sat
  - infinitive–present, e.g. sit–sits
  - infinitive–progressive, e.g. sit–sitting
  - past–present, e.g. sit–sits
  - past–progressive, e.g. sit–sitting
  - present–progressive, e.g. sits–sitting

For English, the dataset was constructed using the word list with inflected forms from the SCOWL project. In the English data, 25,052 nouns, 1,433 adjectives, and 7,806 verbs were used for training. 1000 word pairs were selected randomly for validation and 1000 for testing, evenly distributed among relation types.

Swedish. Words were extracted from SALDO (Borin et al. 2013). In the Swedish data, 64,460 nouns, 12,507 adjectives, and 7,764 verbs were used for training. The division into training, validation, and test sets were based on the same proportions as in English. The same forms were used as in English, except that instead of the progressive form for verbs, the passive infinitive was used, e.g. äta:ätas ‘eat:be eaten’.

Finnish, German, and Russian. For these languages, data from task1 and task2 in SIGMORPHON 2016 was used for training, and task2 data was used for evaluation. In this dataset, each word pair is provided along with morphological tags for the source and target words. We define a relation \( R \) as the combination of two sets of morphological tags, for which there exist words in the data.

The SIGMORPHON datasets consist of word pairs along with the corresponding morphological tags, specifying properties such as gender, number, case, and tense. For training set and validation set, we generate analogies from this as follows. First, we read each word pair

\[ \text{See http://wordlist.aspell.net/}. \]
(\(w_1, w_2\)) from the dataset, building tables of paradigms by storing the
word pairs together with their tags. If \(w_1\) or \(w_2\) with the exact same
set of tags has already been stored in a table (it may be part of another
word pair \((u_1, u_2)\)), then \(w_1\) and \(w_2\) is stored in the same table as \(u_1\)
and \(u_2\). Second, when all words are stored in tables, we go through
them and consider each pair of word forms members of a morphologi-
cal relation. All words having a given source form and target form
make up the set of word pairs for that relation. This procedure allows
us to get more training data, as some new pairs can be generated from
the tables (e.g. \((w_1, u_1)\) from the example above). For the test set, we
do not enhance the data in any such way, but use the exact relations
provided from the original SIGMORPHON data set.

As the task described in this paper differs from the original SIG-
MORPHON task, with the additional requirement that every query–
target word pair needs to be accompanied by a demo relation with
the same forms, all relations with only one word pair were discarded.
Of the SIGMORPHON datasets, we did not include Arabic, Georgian,
Hungarian, Maltese, Navajo, Spanish, and Turkish, either because of
the sparsity problem mentioned above, or because the morphological
features used in the language made it difficult to generate query–target
pairs. The percentage of test set word pairs from the SIGMORPHON
data being discarded in the remaining languages: Finnish: 1.2%, Ger-
man: 2.1%, and Russian: 0.5%. Details about dataset sizes can be found
in Table 2.

4.3 Evaluation

To evaluate the performance of the model, the datasets for English and
Swedish were randomly split into training, validation, and test sets.
Exact dataset split is defined within the openly shared code reposi-
tory. For the SIGMORPHON languages (Finnish, German, and Rus-
sian), the provided dataset split was used, and the test was performed
as specified in the dataset, ignoring the specified morphological tags.
For English and Swedish, each word pair was tested in both directions
(switching the query word and the target word). Within one relation
type, each word pair was randomly assigned another word pair as

---

7 We decided on a threshold of at most 3% of the word pairs that could be
discarded for the evaluation to be meaningful.

8 https://github.com/olofmogren/char-rnn-wordrelations/
Table 2: Number of relations (“Rels”, after discarding size-1 relations) and word pairs (“WPs”) in the data set. *English and Swedish word pairs are all used exactly twice, once in original order, and once reversed. This means that the effective number of word pairs for these two languages are double the numbers in this table.

<table>
<thead>
<tr>
<th>Language</th>
<th>Training set</th>
<th>Validation set</th>
<th>Test set</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rels</td>
<td>WPs</td>
<td>Rels</td>
<td>WPs</td>
</tr>
<tr>
<td>English</td>
<td>10</td>
<td>74,187</td>
<td>10</td>
<td>1,000</td>
</tr>
<tr>
<td>Finnish</td>
<td>1,291</td>
<td>49,312</td>
<td>427</td>
<td>1,246</td>
</tr>
<tr>
<td>German</td>
<td>1,571</td>
<td>58,651</td>
<td>400</td>
<td>1,174</td>
</tr>
<tr>
<td>Russian</td>
<td>830</td>
<td>51,939</td>
<td>285</td>
<td>1,399</td>
</tr>
<tr>
<td>Swedish</td>
<td>10</td>
<td>146,551</td>
<td>10</td>
<td>1,000</td>
</tr>
</tbody>
</table>

demo relation. Each word pair was used exactly once as a demo relation, and once as a query–target pair. Both word pairs in each analogy were selected from the same data partition; i.e. the test set for the evaluation. Relations having only one word pair were dropped from the test set, this is the only difference between the original SIGMORPHON test data and the test data used here (for more information, see Section 4.2). Where nothing else is specified, reported numbers are the prediction accuracy. This is the fraction of predictions that exactly match the target words.

4.4 Data ambiguity

As noted in Section 1, different words can have different inflectional patterns, and some words may also have the same expression for several forms. We note that there are such examples in the training data and in the validation data, but no such examples were detected in any of the test sets, see Table 3. This may affect the training, but not the evaluation of the system. When such ambiguities are presented as the target in demo relations, there is of course no way for a system with this setup to know which form to pick. However, the aim of our study was to keep the setup realistic, and hence, such ambiguous expressions were not removed from the datasets. Since no ambiguities were found in the test sets, there is always exactly one correct target for each query, but with a corresponding amount of
Table 3: Number of ambiguities detected in the data. This is the number of words that occur twice or more as a source word or as a target word in respective dataset partition

<table>
<thead>
<tr>
<th>Language</th>
<th>Training set</th>
<th>Validation set</th>
<th>Test set</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Twice More</td>
<td>Twice More</td>
<td>Twice More</td>
<td>Twice or more</td>
</tr>
<tr>
<td>English</td>
<td>407 6</td>
<td>0 0</td>
<td>0 0</td>
<td>413</td>
</tr>
<tr>
<td>Finnish</td>
<td>20 0</td>
<td>0 0</td>
<td>0 0</td>
<td>20</td>
</tr>
<tr>
<td>German</td>
<td>415 0</td>
<td>2 0</td>
<td>0 0</td>
<td>417</td>
</tr>
<tr>
<td>Russian</td>
<td>186 0</td>
<td>0 0</td>
<td>0 0</td>
<td>186</td>
</tr>
<tr>
<td>Swedish</td>
<td>2,852 41</td>
<td>0 0</td>
<td>0 0</td>
<td>2,893</td>
</tr>
</tbody>
</table>

Table 4: Prediction accuracy and average Levenshtein distance of the proposed model (NMAS) trained using one language. Baseline: Lepage (1998)

<table>
<thead>
<tr>
<th>Language</th>
<th>Accuracy</th>
<th>AVG Levenshtein</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NMAS</td>
<td>Lepage</td>
</tr>
<tr>
<td>English</td>
<td>94.85%</td>
<td>56.05%</td>
</tr>
<tr>
<td>Finnish</td>
<td>85.64%</td>
<td>31.39%</td>
</tr>
<tr>
<td>German</td>
<td>87.96%</td>
<td>76.63%</td>
</tr>
<tr>
<td>Russian</td>
<td>75.85%</td>
<td>48.19%</td>
</tr>
<tr>
<td>Swedish</td>
<td>91.40%</td>
<td>64.80%</td>
</tr>
</tbody>
</table>

ambiguous data in the training set, the model may learn robustness and the noise provided by the ambiguities may also help to regularize the training.

5 RESULTS

This section presents the results of the experimental evaluation of the system.

5.1 Language-wise performance

The prediction accuracy results for the test set can be seen in Table 4, reaching an accuracy of 94.85% for English. While Finnish is a morphologically rich language, with 1323 distinct relations in the dataset, and with the lowest Lepage baseline score of all evaluated languages
The Neural morphological analogy system

(31.39%), NMAS is able to learn its relations rather well, with a prediction accuracy of 85.64%. For German and Swedish, the performance is 87.96% and 91.40%, respectively. They both have more complex morphologies with more inflectional patterns for nouns and verbs. On Russian, NMAS obtains an accuracy of 75.85%. This may be explained by its complex morphology and phonology, and is consistent with the results of top scoring systems on the SIGMORPHON tasks.

5.1.1 Detailed analysis of phonological and orthographic regularities

To successfully predict inflected word forms, our model must take the inflectional paradigms into account, as well as the orthographic and the phonological regularities that sometimes cut across the paradigms. We will now consider a number of examples of such regularities for all five languages and investigate how well they are handled by our model.

English. A basic textbook example of an orthographic rule conditioned on the immediate context is the $y/ie$ alternation in English, such as *fry/fries, hurry/hurried*, etc. This simple regularity poses no difficulty for our model which predicted the correct form in all cases where such alternations occurred in the data.

Finnish. A more interesting case is the phonology of Finnish, which is well-known for its vowel harmony and consonant gradation. To investigate how well the model handles vowel harmony, we considered instances where the final vowel (which typically corresponds to the inflection) in the gold-standard output is either in the back-vowel group ($a$, $o$, or $u$) or the front-vowel group ($ä$, $ö$, or $y$). In these cases, the model predicts the correct vowel of the output form in 97% of the cases. This figure is identical for the subset of instances where vowel in the output is in a different group from the corresponding vowel in the demo output, which occurs in about 18% of the instances. It is notable that several of the model’s vowel harmony errors correspond to exceptions where the usual rules of vowel harmony do not apply: (1) in a compound such as *hiuspinnit* (‘hairpins’), the model is confused by the back vowel $u$ in the compound prefix, and incorrectly predicts an $a$ instead of $ä$ in inflections; (2) a non-native word such as *desiderativi* (‘desiderative’) may take an “unexpected” vowel (in this case the inflections use front vowels despite the preceding $a$).
In consonant gradation, consonants or consonant clusters may appear in either the strong or the weak grade, depending on the phonological context. For instance, the cluster *rt* in *parta* (‘beard’ in the nominative singular) is in the strong grade, which in the weak grade becomes *rr*, as in *parrat* (nominative plural). We selected the occurrences where the query word and the gold-standard output differed in grade (about 15% of the instances). The correct grade is predicted by the model in 84% of these cases, and this does not seem to be affected by whether the demo pair involves consonant gradation or not.

German. For German, we considered umlaut alternations such as *Baum/Bäume*. We selected the instances where there is an umlaut vowel in either the query word or in the gold-standard output, but not both. This is about 2.7% of the instances. In such cases, the model predicts the vowel correctly just 27% of the time. This can be contrasted with results we saw for Finnish, where vowel harmony was handled almost perfectly. This difference is probably due to the unpredictability and rarity of the German umlaut, while the Finnish vowel harmony is very common and almost perfectly regular.

Swedish. Some words in Swedish exhibit the process of syncope where the unstressed vowel *e* is lost in some contexts. For instance, the word *nyckel* (‘key’) has the plural form *nycklar*. We found 36 instances involving a syncope of the query word or the gold-standard output. This corresponds to 1.8% of the total set. The model predicts the correct form in 94% of these cases.

Russian. This is the language which has proven to be the most problematic for our model, due to the rich system of its inflectional paradigms, as well as the complex phonological processes (e.g. palatalization) affecting some of the inflections. While irregularities are challenging for the model, it successfully handles phenomena that are more predictable. As an example of a regular pattern that the model handles well, we can consider the clitic used with reflexive verbs, which takes the form -*sy* or -*s*’ depending on the phonological context. The form of this clitic is predicted correctly by our model 98% of the time.
5.2 Model variants

Attend to relation. Kann and Schütze (2016) explicitly feeds the morphological tags as special tokens being part of the input sequence, and the attention mechanism learns when to focus on the forms during output generation. Inspired by this we decided to evaluate a variant of our model where the embedding of the relation encoder is appended to the query encoder output sequence, allowing the decoder to attend to the whole query as well as the relation embedding, see Figure 2. The per-
Table 5: Prediction accuracy of the proposed model trained with “attend to relation”, with and without the relation embedding fed to initial hidden state (FC combined), all words reversed, feeding the relation embedding using a shortcut to each step in the decoder RNN, and using auxiliary tags classification criterion, respectively. English validation set

<table>
<thead>
<tr>
<th>Variant</th>
<th>Validation accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full model</td>
<td>96.40%</td>
</tr>
<tr>
<td>Attend to relation</td>
<td>96.20%</td>
</tr>
<tr>
<td>Attend to relation &amp; No FC combined</td>
<td>95.35%</td>
</tr>
<tr>
<td>Reversed words</td>
<td>96.00%</td>
</tr>
<tr>
<td>Relation shortcut</td>
<td>95.40%</td>
</tr>
<tr>
<td>Auxiliary tags classification</td>
<td>95.25%</td>
</tr>
</tbody>
</table>

Performance of the model was not affected by this change (see Table 5), and there was no clear trend spanning over different languages. When also disabling FC combined, and thus feeding the relation embedding directly as input to the decoder, there was a noticeable decrease in performance: 95.35% accuracy on the English validation set.

Relation shortcut. In the layout of the proposed model, the information from the relation encoder is available to the decoder only initially. To explore if it would help to have the information available at every step in the decoding process, a shortcut connection was added from FC relation to the final layer in the decoder. This helped the model to start learning fast (see Figure 3), but then resulted in a slight decrease in accuracy (95.25% on English validation set). (See Table 5).

Auxiliary training criterion. Multi-task learning using a related auxiliary task can lead to stronger generalization and better regularized models (Caruana 1998; Collobert and Weston 2008; Bingel and Søgaard 2017). We evaluated a model that used an auxiliary training task: the model had to predict the morphological tags as an output from the relation encoder. This addition gave a slight initial training speedup (see Figure 3), but did not give a better performing model once the model finished training. This indicates a strength in the originally proposed solution: the model can learn to differentiate the morphological forms of the words in the demo relation, even
without having this explicit training signal, something that is also demonstrated by the visualized relation embeddings (see Figure 4).

Disabling model components. The relation encoder learns to represent the different morphological relations with nicely disentangled embeddings (see Figure 4). The fact that the prediction accuracy drops as far as to 39.35% when disabling the relation input (see Table 6) indicates that the setup is useful, and that the model indeed learns to utilize the information from the demo relation. Disabling only the first word in the demo relation allows the model to perform much better (94.60% validation accuracy), but it does not reach the accuracy of the full model with both demo words (96.40%). Disabling the attention mechanism is a small modification of our model, but it substantially degrades performance, resulting in 91.90% accuracy on the English validation set.

5.3 Mechanisms of word inflection

As English (and many other languages) forms inflections mainly by changing suffixes, an experiment was performed where every word was reversed (e.g. “requirement” → “tnemeriuquer”), to evaluate
Figure 4: t-SNE visualization of all demo relation pairs from English validation set embedded using the relation encoder. Each point is colored by the relation type that it represents.

Table 6: Prediction accuracy of the proposed model without attention mechanism, without the first (source) word in the demo relation, and completely without demo relation encoder, respectively. English validation set

<table>
<thead>
<tr>
<th>Variant</th>
<th>Validation accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full model</td>
<td>96.40%</td>
</tr>
<tr>
<td>Disable attention mechanism</td>
<td>91.90%</td>
</tr>
<tr>
<td>Disable relation source</td>
<td>94.60%</td>
</tr>
<tr>
<td>Disable relation input</td>
<td>39.35%</td>
</tr>
</tbody>
</table>

whether the model can cope with other mechanisms of word inflection. On this data, NMAS obtains a prediction accuracy that is only slightly worse than the original version (96.00% on English validation set). This indicates that the model can cope with different kinds of inflectional patterns (i.e. suffix and prefix changes). As can be noted in the example outputs (see Table 7), the model does handle several different kinds of inflections (including orthographic variations such as y/ie), and it does not require the demo relation to show
The Neural morphological analogy system

Table 7: Correct (top), and incorrect (bottom) example outputs from the model. Samples from English validation set

**Correct:**

<table>
<thead>
<tr>
<th>Demo word 1</th>
<th>Demo word 2</th>
<th>Query</th>
<th>Target</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>misidentify</td>
<td>misidentifies</td>
<td>bottleneck</td>
<td>bottlenecks</td>
<td>bottlenecks</td>
</tr>
<tr>
<td>obliterate</td>
<td>obliterated</td>
<td>prig</td>
<td>prigged</td>
<td>prigged</td>
</tr>
<tr>
<td>ventilating</td>
<td>ventilates</td>
<td>disorganizing</td>
<td>disorganizes</td>
<td>disorganizes</td>
</tr>
<tr>
<td>crank</td>
<td>cranker</td>
<td>freckly</td>
<td>frecklier</td>
<td>frecklier</td>
</tr>
<tr>
<td>debauchery</td>
<td>debaucheries</td>
<td>bumptiousness</td>
<td>bumptiousnesses</td>
<td>bumptiousnesses</td>
</tr>
</tbody>
</table>

**Incorrect:**

<table>
<thead>
<tr>
<th>Demo word 1</th>
<th>Demo word 2</th>
<th>Query</th>
<th>Target</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>repackage</td>
<td>repackaged</td>
<td>outrun</td>
<td>outran</td>
<td>outrun</td>
</tr>
<tr>
<td>misinformed</td>
<td>misinform</td>
<td>gassed</td>
<td>gas</td>
<td>gas</td>
</tr>
<tr>
<td>julep</td>
<td>juleps</td>
<td>catfish</td>
<td>catfish</td>
<td>catfishes</td>
</tr>
<tr>
<td>cedar</td>
<td>cedars</td>
<td>midlife</td>
<td>midlives</td>
<td>midlifes</td>
</tr>
<tr>
<td>affrays</td>
<td>affray</td>
<td>buzzes</td>
<td>buzz</td>
<td>buzze</td>
</tr>
</tbody>
</table>

the same inflectional pattern as the query word. In fact, often when the system fails, it does so by inflecting irregular words in a regular manner, suggesting that patterns with less data availability poses the major problem.

5.4 **Relation embeddings**

Figure 4 shows a t-SNE visualization of the embeddings from the relation encoder ("FC relation") of all data points in the English validation set. One can see that most relations have been clearly separated into one distinct cluster each, with the exception of two clusters, both containing points from two relations each. The first such cluster contains the two relation types “N: Singular-Plural” and “V: Infinitive-3 Pers. Sg. Present”; both of these are realized in English by appending the suffix -s to the query word. The second cluster contains the relation types “N: Plural-Singular” and “V: 3 Pers. Sg. Present-Infinitive”; both of these are realized by the removal of the suffix -s. It is worth noting that no ex-
Figure 5: t-SNE visualization of all query words from English validation set embedded using the query encoder. Each point is coloured by the relation type that it represents.

Implicit training signal has been provided for this to happen. The model has learned to separate different morphological relations to help with the downstream task.

Similar clustered representations can be seen when analysing the embeddings computed by the relation encoder RNN also for other languages. We refer interested readers to the supplemental material⁹ for a complete list of these plots.

Figure 5 shows a t-SNE visualization of the embeddings from the query encoder. As we saw with the relation encoder, query embeddings seem to encode information about morphology as similar morphological forms cluster together, albeit with more internal variation and more inter-cluster overlaps. The task for the query encoder is more complex as it needs to encode all information about the query word and provide information on how it may be transformed. To solve the task, and be able to correctly transform query words with the same relation type but with different inflection patterns, it needs to be able to deduce what subcategory of a relation a given query word belongs to.

The Neural morphological analogy system

5.5 Word embedding analogies

The Lepage baseline proved to be the strongest baseline for all languages. For instance, for English it obtains prediction accuracy of 56.05%, compared to 40.75% for the Word2Vec baseline, and 45.00% for the FastText baseline. Without the Lepage fallback, the Word2Vec baseline scored 14.45%, and the FastText baseline scored 22.75%. For other languages, the results were even worse. The datasets in our study contain a rather large vocabulary, not only including frequent words. While the fixed vocabulary is one of the major limitations (explaining the difference between the embedding baselines and the corresponding ones without fallback), the word embedding baseline predictions were often incorrect even when the words were included in the vocabulary. This led us to use the Lepage baseline in the result tables.

5.6 Relation-wise performance

Figure 6 shows the performance for each relation type, showing that our model obtains 100% test set accuracy for the transforms between comparative–superlative. It obtains the lowest accuracy (85.19%) for past–infinitive, 94.17%, and 92.23% for plural–singular, and singular–plural, respectively. From Figure 4 we have learned that these very relations are the most difficult ones for the relation encoder to distin-

Figure 6: Results for all relations (total), and for each specific relation of the English test set
guish between. One difficulty of plural–singular seems to be to determine how many character to remove, while the patterns for adding the -s suffix is generally simpler. An example demonstrating this can be seen in Table 7: buzzes:buzz, where the model incorrectly predicted buzze.

5.7 Example outputs

We have collected some examples from the English validation set where our model succeeds and where it fails (see Table 7). Examples of patterns that can be observed in the failed examples are (1) words with irregular inflections that the model incorrectly inflects using regular patterns, e.g. outrun:outran, where the model predicted outrunned; (2) words with ambiguous targets, e.g. gassed:gas, where the model predicted gass. If there existed a verb gass, it could very well have been gassed in its past-tense form. Tables with example output for the other studied languages are provided in the supplemental material. In general: the model can learn different inflectional patterns. Suffixes, infixes, and prefixes do not pose problems. The query word does not need to have the same inflectional pattern as the demo relation. When the model does fail, it is often due to an inflection that is not represented in the training data, such as irregular verbs.

6 DISCUSSION AND CONCLUSIONS

In this paper, we have presented a neural model that can learn to carry out morphological relational reasoning on a given query word \( q \), given a demo relation consisting of a word in two different forms (source form and desired target form). Our approach uses a character based encoder RNN for the demo relation words, and one for the query word, and generates the output word as a character sequence. The model is able to generalize to unseen words as demonstrated by good prediction accuracy on the held-out test sets in five different languages: English, Finnish, German, Russian, and Swedish. It learns representations that separate the relations well provided only with the training signal given by the task of generating the words in correct form.

\[10\] http://bit.ly/2oyPExT
The Neural morphological analogy system

Our solution is more general than existing methods for morphological inflection and reinflection, in the sense that they require explicit enumeration of the morphological tags specifying the transformation; our solution instead learns to build its own internal representation of this information by observing an analogous word pair demonstrating the relation.

ACKNOWLEDGMENTS

RJ was supported by the Swedish Research Council under grant 2013–4944. OM was supported by Swedish Foundation for Strategic Research (SSF) under grant IIS11-0089.

REFERENCES


Grzegorz CHRUPALA, Georgiana DINU, and Josef VAN GENABITH (2008), Learning morphology with Morfette, in *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08)*, pp. 2362–2367, European Language Resources Association (ELRA), Marrakech, Morocco.


Ryan Cotterell, Hinrich Schütze, and Jason Eisner (2016b), Morphological smoothing and extrapolation of word embeddings, in *Proceedings
The Neural morphological analogy system


Dedre Gentner, Keith James Holyoak, and Boicho N. Kokinov (2001), The analogical mind: Perspectives from cognitive science, MIT press.


Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush (2016), Character-aware neural language models, in Proceedings of the Thirtieth AAAI
Olof Mogren, Richard Johansson

Conference on Artificial Intelligence, AAAI’16, pp. 2741–2749, AAAI Press, Phoenix, United States.


The Neural morphological analogy system


*This work is licensed under the Creative Commons Attribution 3.0 Unported License.*

http://creativecommons.org/licenses/by/3.0/