Character-based recurrent neural networks for morphological relational reasoning

Olof Mogren
Chalmers University of Technology
Sweden
mogren@chalmers.se

Richard Johansson
University of Gothenburg
Sweden
richard.johansson@gu.se

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 95.60%. 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.

1 Introduction

Analogical reasoning is an important part of human cognition (Gentner et al., 2001). Resolving analogies by mapping unknown data points to known analogous examples allow 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. 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. To gain insight about how systems can learn the right abstraction using limited supervision to generate inflected words, is important for how to create systems for more complex language generation tasks, such as machine translation, automatic summarization, and dialog 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 to this problem 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 orthograpic and phonological rules, and is is also 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, these 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 plays 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 the three input words, determining changes in prefixes, infixes, and suffixes, and adding or removing them to or from the the query word, transforming it into the target:
reader:unreadable = *doer:***x** → **x** = *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 reinflection* (Cotterell et al., 2016a; Cotterell et al., 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., 1992), 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 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 system from one source of errors. Neural models working on character- or subword-level have been applied in several NLP tasks, ranging from relatively basic tasks such as text categorization (Zhang et al., 2015) and language modeling (Kim et al., 2016) to complex prediction tasks such as translation (Luong and Manning, 2016; Sennrich et al., 2016). Because they can recognize patterns 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 (Koskenniemi, 1984; Kanis and Müller, 2005; Chrupała et al., 2008; Jongejan and Dalianis, 2009; Chakrabarty et al., 2017). In comparison, our task does not offer contextual information, but instead provides the (similarly implicit) cues for forms from the demo relation. With this in mind, predicting the lemma is just a special case of the morphological 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 utility of, and to evaluate the quality of neural word embeddings (Mikolov et al., 2013a; Mnih and Kavukcuoglu, 2013; Nicolai et al., 2015b; Pennington et al., 2014). Such embeddings show simple linear relationships in the resulting continuous embedding space that allow for finding impressive analogous relations such as

$$v(king) - v(man) + v(woman) \approx v(queen).$$

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 Neural Morphological Analogies System

In this paper, we present the Neural Morphological Analogies System (NMAS), a neural approach for morphological relational reasoning. We use a deep recurrent neural network with 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

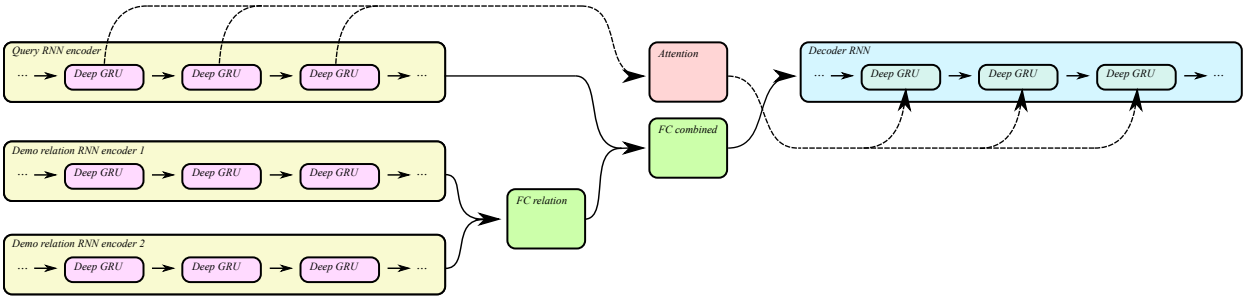

Figure 1: The layout of the proposed model. The demo relation is encoded using two encoder RNNs with shared weights for the two demo word forms. A fully connected layer *FC relation* follows the demo relation pair. The query word is encoded separately, and its embedding is concatenated with the output from *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.

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 f_t), what parts of the input vector to store in the internal memory (the input gate i_t), and what to include in the output vector (the output gate o_t). The Gated Recurrent Unit (GRU) (Cho et al., 2014a) is a simplification of this approach, having only two gates by replacing the input and forget gates with an update gate u_t 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 directly on the raw character stream, 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 $R_{demo} = (w_1, w_2)$ 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*:

$$\mathbf{h}_{rel} = tanh(W_{rel}[g_{rel}(\mathbf{0}, \mathbf{w}_1), g_{rel}(\mathbf{0}, \mathbf{w}_2)]),$$

where g_{rel} is the output from the relation encoder RNN (using zero vectors as initial hidden states), $\mathbf{w}_1, \mathbf{w}_2$ are sequences of one-hot encodings for the characters of w_1 and w_2, W_{rel} is the weight matrix for the *FC relation* layer, and $tanh$ is the element-wise nonlinearity. Here, $[\mathbf{x}, \mathbf{y}]$ means the concatenation of the vectors $\mathbf{x}$ and $\mathbf{y}$.

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*:

$$\mathbf{h}_{comb} = tanh(W_{comb}[\mathbf{h}_{rel}, g_q(\mathbf{0}, \mathbf{q})]),$$

where $\mathbf{h}_{rel}$ is the output from *FC relation*, g_q is the output from the query RNN encoder, $\mathbf{q}$ is a sequence of one-hot encodings of the chracters of the query word, W_{comb} is the weight matrix for the *FC combined* layer, and $tanh$ is the element-wise nonlinearity. The result $\mathbf{h}_{comb}$ 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 $p(y_{t_d} = i) = \mathbf{h}^{(i)}_{dec;t_d}$ for each character i in the alphabet and for each step t_d in the decoder is modeled using:

$$\mathbf{h}_{dec;t_d} = s(W_{dec}[g_{dec}(\mathbf{h}_{comb}, \mathbf{y}_{(0:t_d-1)}), \mathbf{a}]),$$

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

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.

4.1 Implementation details

The model was implemented using PyTorch;[1] all source code is freely available.[2] With the final hyperparameter settings (see Section 4.2), the model contains approximately 155000 parameters, and it can be trained in a few hours on a modern GPU.

4.2 Hyperparameters

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

Hyperparameter	Explored	Selected
Embedding size	50-350	100
Hidden size	25-350	50
FC relation size	50-350	50
FC combined size	50-350	100
RNN depth	1-3	2
Learning rate		1×10^{-3}
L2 weight decay		5×10^{-5}
Drop probability	0.0, 0.2, ..., 0.8	0.0

Table 1: Hyperparameters in the model.

4.3 Training

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 realation type; one of these will be the demo relation, and one will be the query–target pair. The output from the decoder (see Section 3.3 C), is a categorical distribution over the alphabet. We use the cross-entropy loss function for

[1] http://pytorch.org/

[2] http://mogren.one/

Language	Training set		Validation set		Test set		Total	
	Rels	WPs	Rels	WPs	Rels	WPs	Rels	WPs
English	10	74187	10	1000	10	1000	10	76187*
Finnish	1293	50269	431	1255	1092	11471	1323	62995
German	1572	70429	421	1271	1249	7768	1572	79468
Russian	1001	56119	290	1421	666	11492	1003	69031
Swedish	10	146551	10	1000	10	1000	10	148551*

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.

each character at position t_d in the output word as the learning objective for all parameters θ in the model:

$$\mathcal{L}(\theta) = \frac{-\sum_{t_d} \mathbf{y}_{(t_d)} \cdot \log \mathbf{h}_{\theta;t_d}}{N},$$

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 $\mathbf{h}_{\theta;t_d}$ 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.

4.4 Baselines

The task considered in this work is closely related to morphological reinflection, and systems for this 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 (W2V) (Mikolov et al., 2013b) and FastText (FT) (Bojanowski et al., 2016). 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

$$v(q) - v(w_1) + v(w_2),$$

where $v(w)$ 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 Word2Vec CBOW embeddings used were downloaded from Kyubyong Park's repository.[3]. The embeddings were trained using data from Wikipedia. The FastText embeddings used were downloaded from the FastText authors' website.[4] 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.

[3] https://github.com/Kyubyong/wordvectors/

[4] https://fasttext.cc/

	Accuracy		AVG Levenshtein	
	NMAS	**Lepage**	**NMAS**	**Lepage**
English	95.60%	56.05%	0.06	0.67
Finnish	83.26%	31.39%	0.25	1.76
German	89.12%	76.63%	0.18	0.39
Russian	70.54%	48.19%	0.45	1.01
Swedish	90.10%	64.80%	0.16	0.60

Table 3: Prediction accuracy and average Levenshtein distance of the proposed model (NMAS) trained using one language. Baseline: Lepage (1998).

4.5 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). Code for downloading the data and performing dataset split will be made available when this paper is published.

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

- nouns:
  - singular–plural, e.g. *dog–dogs*
- 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 froms from the SCOWL project.[5] 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.

[5]See `http://wordlist.aspell.net/`.

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. We generate analogies from this as follows. Firstly, we read each word pair (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 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. Secondly, when all words are stored in tables, we go through them, and consider each pair of word forms members of a *morphological relation*. All words having a given source form and target form make up the set of words for that relation.

As the task described in this paper differs from the original SIGMORPHON 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,[6] or because the morphological features used in the language made it difficult to generate query–target pairs. The percentage of test set word pairs being discarded in the remaining languages: Finnish: 1.2%, German: 2.1%, and Russian: 0.5%. Details about dataset sizes can be found in Table 2.

4.6 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 will be distributed when paper is deanonymized. For the SIGMORPHON languages (Finnish, German, and Russian), 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 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 was selected from the same data partition; i.e. the test set for the evaluation. Relations having only one word pair was 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 4.5). Where nothing else is specified, reported numbers are the prediction accuracy. This is the fraction of predictions that exactly match the target words.

4.7 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. 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 was retained in the dataset. The existence of such forms in the test set may put a bound on the achievable performance, but with a corresponding amount of ambiguous data in the training set, the model will learn robustness and the noise provided by the ambiguities may also help to regularize the training.

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

Variant	Accuracy
Disable attention	89.30%
Disable relation source	93.55%
Disable relation input	37.30%

Table 4: 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 test set. Size: 50.

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 3, reaching an accuracy of 95.60% 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 (31.39%), NMAS is able to learn its relations rather well, with a prediction accuracy of 83.26%. For German and Swedish, the performance is 89.12%, and 90.10%, respectively. They both have more complex morphologies with more inflectional patterns for nouns and verbs. On Russian, NMAS obtains an accuracy of 70.54%. This may be explained by its complex morphology and phonology, and is consistent with the results of top scoring systems on the SIGMORPHON tasks.

5.2 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 re-

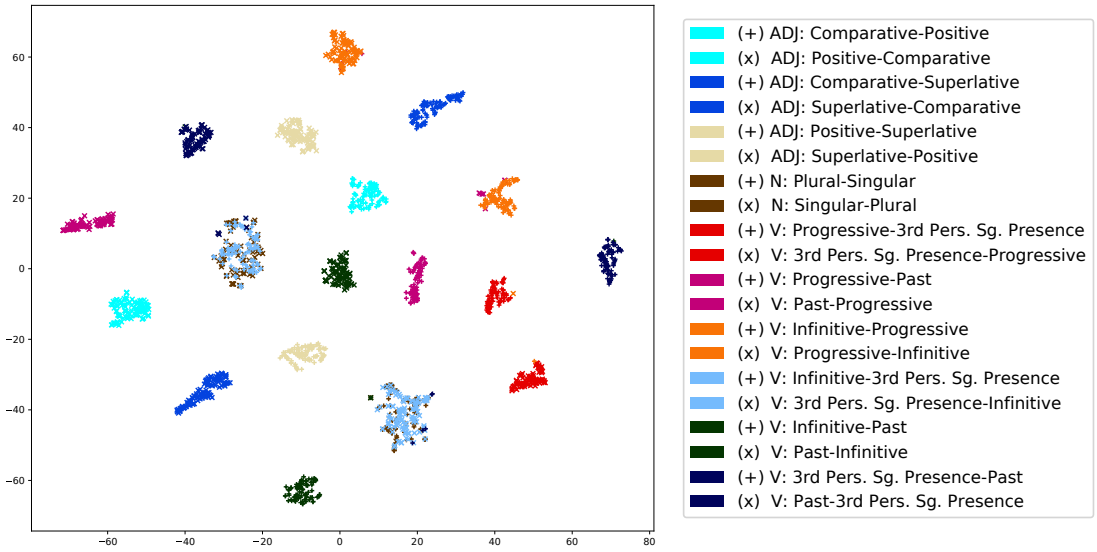

Figure 2: t-SNE visualization of all demo relation pairs from English validation set embedded using the relation encoder. Each point is colored by the relation type that it represents.

Variant	Accuracy
Attend to relation	95.50%
Attend to relation & No *FC combined*	94.75%
Reversed words	94.10%
Relation shortcut	94.45%
Auxiliary tags classification	93.85%

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 test set. Size: 50.

lation embedding. The performance of the model changed minimally by this change (see Table 5), and there was no clear trend spanning over different languages. When also disabling *FC combined*, and thus the relation embedding input to the decoder, there was a noticeable decrease in performance: 94.75% accuracy on the English test 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 4), but then resulted in a slight decrease in accuracy (94.45% on English test 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 4), 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 2).

Disabling model components. The relation encoder learns to represent the different morphological relations with nicely disentangled embeddings (see Figure 2). The fact that the prediction accuracy drops as far as to 37.30% when disabling the relation input (see Table 4) 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

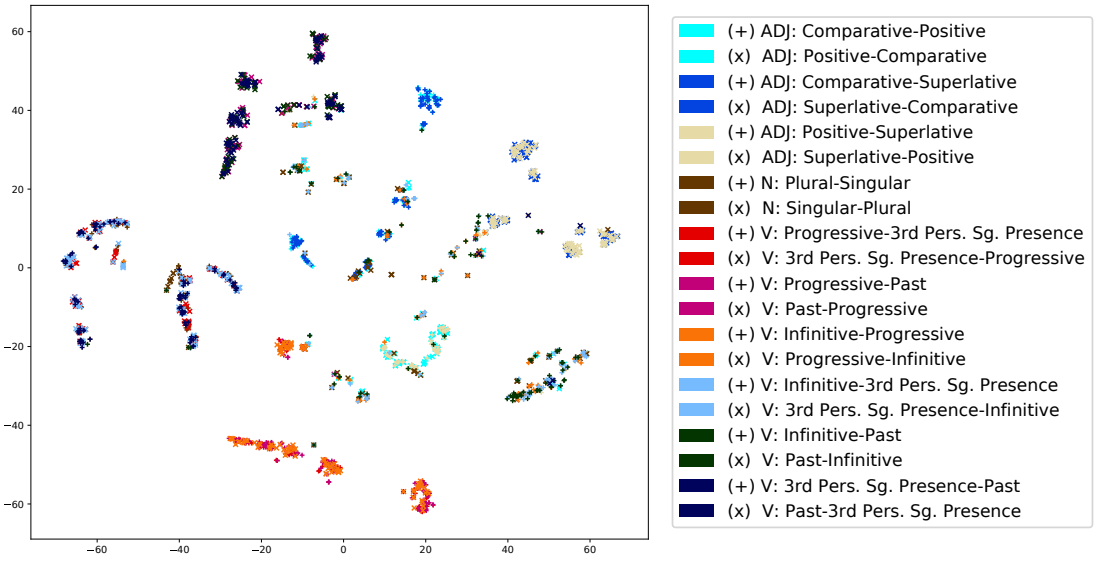

Figure 3: t-SNE visualization of all query words from English validation set embedded using the query encoder. Each point is colored by the relation type that it represents.

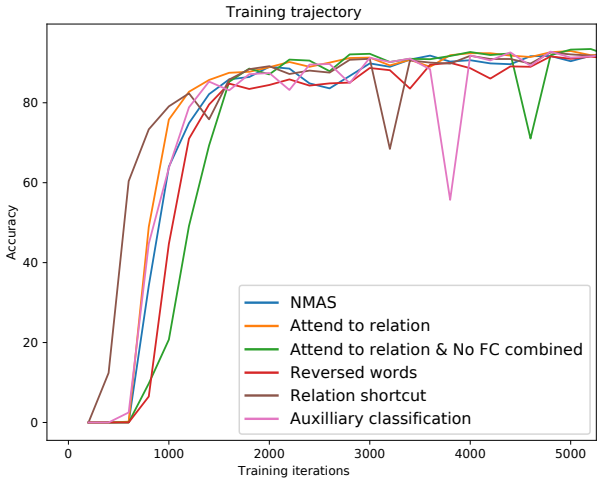

Figure 4: Prediction accuracy on the English validation set during training for some variations of the model.

to perform much better (93.55% accuracy), but it does not reach the accuracy of the full model with both demo words (95.60%). Disabling the attention mechanism is a small modification of our model, but also substantially degrades performance, resulting in 89.30% accuracy on the English test 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* → *tnemeriuqer*), to evaluate 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. 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 6), 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 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 2 shows a t-SNE visualization of the embeddings from the relation encoder (*"FC relation"*) of all datapoints 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

Correct:				
Demo word 1	**Demo word 2**	**Query**	**Target**	**Output**
misidentify	misidentifies	bottleneck	bottlenecks	bottlenecks
obliterate	obliterated	prig	prigged	prigged
ventilating	ventilates	disorganizing	disorganizes	disorganizes
crank	cranker	freckly	frecklier	frecklier
debauchery	debaucheries	bumptiousness	bumptiousnesses	bumptiousnesses
Incorrect:				
Demo word 1	**Demo word 2**	**Query**	**Target**	**Output**
repackage	repackaged	outrun	outran	outrunned
misinformed	misinform	gassed	gas	gass
julep	juleps	catfish	catfish	catfishes
cedar	cedars	midlife	midlives	midlifes
affrays	affray	buzzes	buzz	buzze

Table 6: Correct (top), and incorrect (bottom) example outputs from the model. Samples from English validation set.

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

Figure 3 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 the 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.

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 contains 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 5 shows the performance for each relation type, showing that our model obtains 100% test set accuracy for the transforms between *3rd pers. sg present–infinitive*. It obtains the lowest accuracy (91.76%) for *plural–singular*. From Figure 2 we have learned that these very relations are the most difficult ones for the relation encoder to distinguish between. (The inverse *singular–plural* is generally easier; one difficulty of *plural–singular* seem 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 6: *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 6). Examples of patterns that can be observed in the failed examples are (1) words with irregular inflections that the model incorrectly

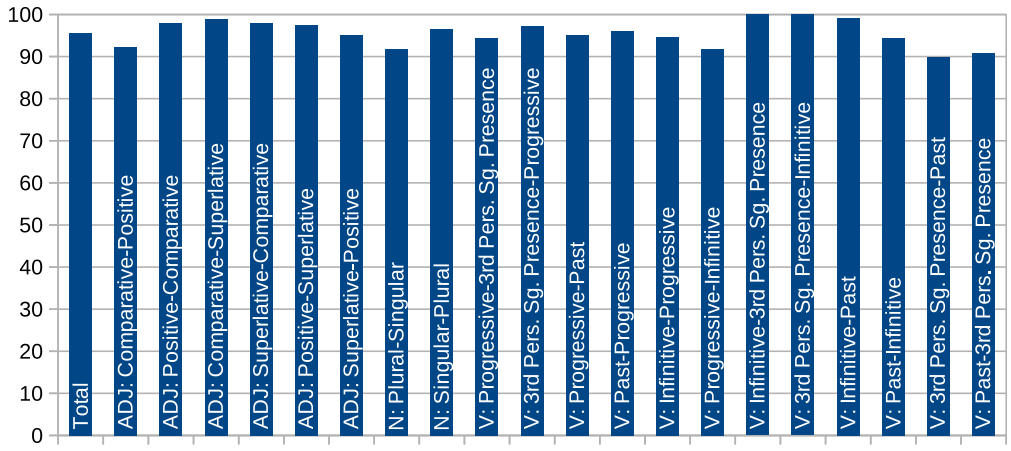

Figure 5: Results for all relations (total), and for each specific relation of the English test set.

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 had 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 is provided in the supplemental material.[7] 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 do 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 the 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.

[7] `http://bit.ly/2oyPEtX`

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

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1024–1029, Denver, Colorado, May–June. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep

neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169, Valencia, Spain, April. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Lars Borin, Markus Forsberg, and Lennart Lönngren. 2013. SALDO: a touch of yin to WordNet's yang. *Language Resources and Evaluation*, 47(4):1191–1211.

Rich Caruana, 1998. *Multitask Learning*, pages 95–133. Springer US, Boston, MA.

Abhisek Chakrabarty, Onkar Arun Pandit, and Utpal Garain. 2017. Context sensitive lemmatization using two successive bidirectional gated recurrent networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1481–1491.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *EMNLP*, pages 1724–1734. ACL.

Grzegorz Chrupała, Georgiana Dinu, and Josef Van Genabith. 2008. Learning morphology with Morfette. In *Proceedings of the 6th edition of the Language Resources and Evaluation Conference*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016a. The SIGMORPHON 2016 shared task – morphological reinflection. In *Proceedings of the 14th Annual SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016b. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1651–1660, Berlin, Germany, August. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30, Vancouver, Canada, August. Association for Computational Linguistics.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a dirichlet process mixture model. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 616–627, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1185–1195, Atlanta, Georgia, June. Association for Computational Linguistics.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of NAACL-HLT*, pages 634–643.

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

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Bart Jongejan and Hercules Dalianis. 2009. Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 145–153, Suntec, Singapore, August. Association for Computational Linguistics.

Jakub Kanis and Luděk Müller. 2005. Automatic lemmatizer construction with focus on oov words lemmatization. In *International Conference on Text, Speech and Dialogue*, pages 132–139. Springer.

Katharina Kann and Hinrich Schütze. 2016. MED: The LMU system for the SIGMORPHON 2016 shared task

on morphological reinflection. In *Proceedings of the 14th Annual SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 62–70.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Kimmo Koskenniemi. 1984. A general computational model for word-form recognition and production. In *Proceedings of the 10th international conference on Computational Linguistics*, pages 178–181. Association for Computational Linguistics.

Yves Lepage. 1998. Solving analogies on words: an algorithm. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 728–734, Montreal, Quebec, Canada, August. Association for Computational Linguistics.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany, August. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June. Association for Computational Linguistics.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2265–2273. Curran Associates, Inc.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015a. Inflection generation as discriminative string transduction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 922–931, Denver, Colorado, May–June. Association for Computational Linguistics.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015b. Morpho-syntactic regularities in continuous word representations: A multilingual study. In *VS@ HLT-NAACL*, pages 129–134.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

David Pesetsky. 2013. *Russian case morphology and the syntactic categories*. MIT Press.

Graeme D Ritchie, Graham J Russell, Alan W Black, and Stephen G Pulman. 1992. Computational morphology.

Jürgen Schmidhuber and Sepp Hochreiter. 1997. Long short-term memory. *Neural computation*, 7(8):1735–1780.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August. Association for Computational Linguistics.

Nicolas Stroppa and François Yvon. 2005. An analogical learner for morphological analysis. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 120–127, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.

C. Wang, S. S. Venkatesh, and J. S. Judd. 1994. Optimal stopping and effective machine complexity in learning. In *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 207–216. Association for Computational Linguistics.

François Yvon. 1997. Paradigmatic cascades: A linguistically sound model of pronunciation by analogy. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 428–435, Madrid, Spain, July. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.