Neural Sequence-to-Sequence Models for Low-Resource Morphology

Katharina Kann

München 2018

Neural Sequence-to-Sequence Models for Low-Resource Morphology

Katharina Kann

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Katharina Kann

München, den 15. Mai 2018

Erstgutachter: Prof. Hinrich Schütze
Zweitgutachter: Prof. David Yarowsky
Drittgutachter: Prof. Ondřej Bojar

Tag der mündlichen Prüfung: 27. März 2019

Eidesstattliche Versicherung
(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. 5.)

Hiermit erkläre ich an Eides statt, dass die Dissertation von mir selbstständig ohne unerlaubte Beihilfe angefertigt wurde.

New York, den 15.05.2018

Katharina Kann

Abstract

The vocabularies of morphologically rich languages, e.g., German or Spanish, are much larger than those of morphologically poor languages, e.g., Chinese. This is due to processes like composition, inflection and derivation. A result of this is an increased number of rare words, which motivates the development of models that can deal with rare words by either analyzing or generating them.

This thesis presents approaches to generate *morphological inflections* or analyze word forms through *canonical segmentation*. We employ state-of-the art deep learning models, namely encoder-decoder gated recurrent neural networks with attention. For analysis, we simply encode the input word and output the final segments as one word with segmentation markers. For generation, instead of only encoding the source form, we further encode additional information about the source and target forms: their morphological tags. This enables us to train one single model for all combinations of source and target tags. Our approach can thus be seen as *multi-task learning* by considering each such combination a separate task. It enables parameter sharing and strongly boosts the overall performance of our models.

In theory, our methods are language independent. However, due to neural networks requiring large amounts of training data, limited availability of suitable training examples is a big challenge. In this work, we focus particularly on low-resource languages. In order to make up for missing training data, we experiment with *transfer learning* to leverage knowledge from related languages. We show that similar languages can improve performance of morphological generation systems in low-resource settings. Further, we investigate *semi-supervised training*, making use of large unlabeled corpora.

Finally, we focus on the fact that in morphologically rich languages paradigms usually consist of more than one form and present solutions for such *multi-source* settings. We show that both our proposed approaches — a model accepting multi-source input and a fine-tuning approach which is related to domain adaptation — improve generation performance.

Zusammenfassung

Das Vokabular morphologisch reicher Sprachen wie Deutsch oder Spanisch ist um einige Ordnungen größer als das morphologisch armer Sprachen wie Chinesisch. Das ist Prozessen wie Komposition, Flexion und Derivation geschuldet. Das Ergebnis ist eine große Menge seltener Wörter, was die Entwicklung von Modellen, die mit diesen umgehen können, indem sie sie entweder analysieren oder generieren, motiviert.

Diese Doktorarbeit präsentiert Methoden zum Generieren von *morphologischen Flexionen* oder zur Analyse von Wortformen durch *kanonische Segmentierung*. Dafür verwenden wir moderne Deep Learning-Modelle, nämlich eine neuronale Architektur, die "encoder-decoder gated recurrent neural network" genannt wird und über einen "attention"-Mechanismus verfügt. Zum Analysieren kodieren wir einfach das Eingabewort und geben alle Segmente als ein Wort mit Markierungen der Segmentgrenzen aus. Zum Generieren kodieren wir nicht nur die Eingabeform sondern noch zusätzliche Informationen über die Eingabe- und Ausgabeform: ihre morphologischen Tags. Das macht es uns möglich, ein einziges Modell für alle Kombinationen aus Quell- und Zieltag zu trainieren. Unser Lösungsansatz kann daher als "multi-task learning" gesehen werden, wenn man jede solche Kombination als einen eigenen Task ansieht. Das macht eine gemeinsame Verwendung der Parameter möglich und verbessert die Gesamtperformanz unserer Modelle stark.

Theoretisch sind unsere Methoden sprachunabhängig. Weil neuronale Netze allerdings große Mengen Trainingsdaten benötigen, ist die begrenzte Verfügbarkeit passender Trainingsbeispiele eine große Herausforderung. In dieser Arbeit konzentrieren wir uns speziell auf Sprachen mit begrenzten Ressourcen. Um fehlende Trainingsdaten auszugleichen, experimentieren wir mit "transfer learning", um Daten verwandter Sprachen zu verwenden. Wir zeigen, dass ähnliche Sprachen im Fall begrenzter Ressourcen die Performanz der Modelle für morphologische Generierung verbessern können. Außerdem untersuchen wir "semi-supervised training", um große, nicht annotierte Korpora zu verwenden.

Schließlich konzentrieren wir uns auf die Tatsache, dass Paradigmen in morphologisch reichen Sprachen normalerweise aus mehr als einer Form bestehen und presentieren Lösungen for solche "multi-source"-Settings. Wir zeigen, dass un-

sere beiden vorgeschlagenen Ansätze, ein Modell, dass mehrere Eingaben akzeptiert und ein ”fine-tuning” Ansatz, der ”domain adaptation” ähnelt, die Performanz der Generierung verbessern.

Acknowledgments

Writing this dissertation was a hard but beautiful process during which I learned about natural language processing and machine learning, the world and its inhabitants, as well as myself. It would by no means have been possible without the support and guidance of a variety of people along the way, and I would like to explicitly mention the most important ones here.

First of all, I would like to thank my advisor Hinrich Schütze for the huge amount of time and all the advice he gave me. He taught me how to conduct interesting research, and shielded me and my fellow PhD students from as much non-research related work as possible. I am grateful that he gave me the opportunity to leave Munich for a total of three internships during the time of my PhD.

Second, I owe a lot to my amazing colleagues and friends at CIS, who I miss every day at 12:00 and 15:00: Anne Friedrich, Ben Roth, Heike Adel, Matthias Huck, Sascha Rothe, Sebastian Ebert, Thang Vu, Wenpeng Yin, and Yadollah Yaghoobzadeh. I promised Heike a tiny PhD hat in case I finished my thesis, so I drew her one here. I will always be grateful for her infinite support, both personally and academically.

Third, I am thankful to my colleagues and friends at Google, most importantly my hosts Adnan Ozturel, Katja Filippova, Ralf Perpeet, and Sascha Rothe, as well as Manu Orsini and Paulina Grnarova.

Finally, I owe all success I have ever achieved to my family and those friends who feel like family: Monika Pfeiffer-Kann, Peter Kann, Philipp Kann, Alexandra Breiner, Anne Strässer, Diego Oller Alcay, Julian Kling, Michael Kirsche, Patsch Renninger, Raphaela Stork, Robin Zenz, and Tomasz Kuswik. **You are my life, and this dissertation is dedicated to you.**

For Heike.

Contents

Publications and Declaration of Co-Authorship

Chapter 2

Chapter 2 corresponds to the following publication:

> Katharina Kann and Hinrich Schütze; **Single-Model Encoder-Decoder with Explicit Morphological Representation for Reinflection**; Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Berlin, Germany, August 2016), pages 555–560.

I regularly discussed this work with my advisor, but I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My advisor assisted me in improving the draft.

Chapter 3

Chapter 3 corresponds to the following publication:

> Katharina Kann and Hinrich Schütze; **MED: The LMU System for the SIGMORPHON 2016 Shared Task on Morphological Reinflection**; Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology (Berlin, Germany, August 2016), pages 62–70.

I regularly discussed this work with my advisor, but I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My advisor assisted me in improving the draft.

There is a large overlap between the SIGMORPHON workshop paper and the ACL 2016 short paper. I include both papers in this dissertation for the following reasons. ACL is the leading conference in our field, so that acceptance at ACL is

an indication of the quality of the work. The ACL paper also contains a detailed description of POET, an edit-tree-based correction method for morphological reinflection systems that is essentially orthogonal to encoder-decoder methods. The SIGMORPHON paper documents my participation in the SIGMORPHON shared task. My system was the winner of the shared task, which in natural language processing is viewed as an important validation of the underlying approach. The SIGMORPHON paper contains more experimental results and more analysis than the ACL paper.

Chapter 4

Chapter 4 corresponds to the following publication:

> Katharina Kann, Ryan Cotterell and Hinrich Schütze; **One-Shot Neural Cross-Lingual Transfer for Paradigm Completion**; Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Vancouver, Canada, August 2017), pages 1993–2003.

Ryan Cotterell and I came up together with the research question addressed in this work. The data used in the experiments was provided by Ryan Cotterell. I also regularly discussed this work with my coauthors. Apart from these explicitly declared exceptions, I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My coauthors assisted me in improving the draft.

Chapter 5

Chapter 5 corresponds to the following publication:

> Katharina Kann and Hinrich Schütze; **Unlabeled Data for Morphological Generation With Character-Based Sequence-to-Sequence Models**; Proceedings of the 1st Workshop on Subword and Character Level Models in NLP (Copenhagen, Denmark, September 2017), pages 76–81.

I regularly discussed this work with my advisor, but I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My advisor assisted me in improving the draft.

Chapter 6

Chapter 6 corresponds to the following publication:

> Katharina Kann, Ryan Cotterell and Hinrich Schütze; **Neural Multi-Source Morphological Reinflection**; Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (Valencia, Spain, April 2017), pages 514–524.

The research question addressed in this work was suggested by Ryan Cotterell. The data used in the experiments was provided by Ryan Cotterell. I also regularly discussed this work with my coauthors. Apart from these explicitly declared exceptions, I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My coauthors assisted me in improving the draft.

Chapter 7

Chapter 7 corresponds to the following publication:

> Katharina Kann and Hinrich Schütze; **The LMU System for the CoNLL-SIGMORPHON 2017 Shared Task on Universal Morphological Reinflection**; Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection (Vancouver, Canada, August 2017), pages 40–48.

I regularly discussed this work with my advisor, but I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My advisor assisted me in improving the draft.

Chapter 8

Chapter 8 corresponds to the following publication:

> Katharina Kann, Ryan Cotterell and Hinrich Schütze; **Neural Morphological Analysis: Encoding-Decoding Canonical Segments**; Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (Austin, USA, November 2016), pages 961–967.

The research question addressed in this work was suggested by Ryan Cotterell. The data used in the experiments was provided by Ryan Cotterell. Further, Ryan

Cotterell did the original implementation of the reranker. I also regularly discussed this work with my coauthors. Apart from these explicitly declared exceptions, I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My coauthors assisted me in improving the draft.

New York, den 15.05.2018

Katharina Kann

Chapter 1

Introduction

1.1 About Morphology

Within the broader area of natural language processing (NLP), morphology is the study of the internal structure of words, or, more specifically, systematic covariation in the form and meaning of words. The covariations of interest typically occur systematically in groups of words in a language.

Traditionally, a distinction is made between inflectional morphology and derivational morphology: **Inflectional morphology** is concerned with changes to a word's surface form in order to express grammatical properties such as case, gender, tense, etc. The base form or dictionary form of a word is called its lemma, and other forms of a lemma are referred to as word forms or inflected forms. The set of all inflected forms of a lemma is called its paradigm. The English inflected form *rains*, for example, contains the information that the subject of the lemma *rain* is 3rd person singular, e.g., *it*, and that the action occurs in the present. Those transformations of surface forms comply with the syntactic function of a word and correspond usually to rather small changes of a word's semantics. In particular, the part of speech remains unchanged. The subfield of morphology that is concerned with the relationship with syntax is called morphosyntax.

In contrast to inflectional morphology, **derivational morphology** covers transformations that mostly convey a larger change in meaning, though the original and the derived form are still semantically related. An example is the noun *rain* and its derived adjective *rainy*. Derivation is a way of word formation, because it produces new lemmas, while inflection generates new forms of an existing lemma. However, the distinction between inflection and derivation is not always well defined.

One big difference between derivational and inflectional morphology is that whenever a certain inflected form exists for one lemma, chances are high that a

parallel form can be found in the paradigm of most other lemmas. However, this is not the case for derivation: While *rainy* is a common English word, *computery* is not a valid derivation of *computer*. Derivational morphology is, thus, considered harder to master than inflectional morphology.

Another type of morphological word formation is **compounding**. Compounding is the process of combining two independent words into one new word, thus fusing the original meanings. The resulting word's part of speech is not required to be the same as the part of speech of the original words. Neither do the two fused words necessarily have to belong to the same part of speech. English examples are *greenhorn* for a combination of an adjective and a noun or *dreamcatcher* for a compound formed by two nouns.

This thesis is devoted to the computational treatment of morphological phenomena in natural language. With a main focus on inflectional morphology, we aim at finding answers to several questions of different granularity: How can words be split up into their smallest meaning bearing units, the morphemes? How can new surface forms be generated given the meaning they should convey and how do proposed approaches generalize for a variety of languages? And, finally, how are word forms organized into structured classes within their inflectional families and how can we make use of that knowledge to generate inflections?

Inspired by recent advances in the deep learning sub-field of machine learning, we present several neural network-based approaches for morphological generation and analysis, which obtain new state-of-the-art results on multiple tasks addressing above questions. However, neural networks are known for requiring large amounts of training data, while, in contrast, for many morphologically rich languages only few or little morphologically annotated resources are available. This makes the development of methods for handling of morphology challenging, and, subsequently, reduces system performance for many downstream NLP tasks. In order to help overcome this problem which many morphologically rich languages are facing, a big focus of this thesis is on how to make state-of-the-art neural models applicable in low-resource settings.

This introduction covers the background needed to understand both the tasks which are the focus of this thesis as well as the machine learning models which are used in the work described in the following chapters. It is structured as follows. We first introduce the tasks addressed in this work: (i) the generation tasks of morphological inflection, morphological reinflection and paradigm completion; as well as (ii) canonical segmentation, an analysis task. We then explain the neural network architectures that are used in the main part of this thesis: perceptrons, recurrent neural networks (RNNs) and recurrent neural sequence-to-sequence architectures. When the reader is familiar with the problems we are intending to solve, as well as the architectures we use, we present the most important previous approaches for both categories of tasks. Finally, we summarize the core ideas

presented in subsequent chapters of this work.

1.2 Morphological Tasks

1.2.1 The Importance of Handling Morphology

Morphologically rich languages like German, Navajo or Arabic make extensive use of inflection. This means they modify the surface form of a word in order to express grammatical properties like tense or mood of verbs or number or gender of nouns. Many languages realize those modifications through affixiation: they combine different prefixes or suffixes with the stem of a word. However, this is not a general rule, e.g., templatic languages such as Arabic or Maltese apply specific patterns to a word's root instead. In the case of Arabic, the root of a word usually consists of 3 consonants (Sakakini et al., 2017).

Paradigm sizes differ between languages, cf. Table 1.1 for an example of a German noun and Table 1.2 for an example of a Slovak noun. As can be seen, the Slovak nominal paradigm consists of 12 forms, while the German nominal paradigm only covers 8 forms. Additionally, the sizes of paradigms of different parts of speech of one single language usually differ as well: Spanish verbal paradigms are relatively big, while Spanish nominal paradigms only consist of two forms.

	SG	PL
NOM	Schneemann	Schneemänner
GEN	Schneemannes	Schneemänner
DAT	Schneemann	Schneemännern
ACC	Schneemann	Schneemänner

Table 1.1 – *The complete paradigm of the German masculine noun* Schneemann *(Engl.:* snowman*).*

Those transformations of surface forms in morphologically rich languages yield a very large vocabulary. Here, we only show a German and a Slovak nominal paradigm, which are both relatively small. However, the paradigm of a Russian verb, for example, consists of more than 30 different members (Wade, 2010). Polish is even more extreme, since each Polish verb can have about 100 inflected forms (Janecki, 2000).

Thus, inflection as well as other morphological processes result in a drastically increased amount of potentially rare surface forms, which masks important generalizations. Accordingly, morphologically rich languages pose a challenge to computational approaches due to sparsity; a challenge which can be simplified by

	SG	PL
NOM	syn	synovia
GEN	syna	synov
DAT	synovi	synom
ACC	syna	synov
INS	synom	synmi
ESS	synovi	synoch

Table 1.2 – *The complete paradigm of the Slovak noun* syn *(Engl.:* son*).*

special analysis or generation of surface forms. Therefore, morphological treatment can yield an improved performance in many downstream tasks, e.g., speech recognition (Hirsimäki et al., 2006), machine translation (De Gispert et al., 2009; Green and DeNero, 2012) or word representation learning (Avraham and Goldberg, 2017). In particular, some form of morphological generation is trivially needed for all tasks that include the production of human-readable text in morphologically rich languages.

1.2.2 Morphological Generation

In this thesis, we consider a group of closely related tasks concerned with the generation of inflected forms[1]: We define **morphological reinflection** as the task of producing an inflected form from a paradigm, given the lemma or any other inflected form. Note that as the task consists of producing indicated target forms, this does not necessarily include the generation of entire paradigms. A special case of this is **morphological inflection**. Instead of an arbitrary one of the paradigm's forms, the lemma is defined to be the input to the morphological generation system. Finally, we consider the task of **paradigm completion**. Here the goal is to generate *all* inflected forms of a paradigm, i.e., to produce the *entire* paradigm. Either one single form or multiple forms can be given as input. This differs from morphological reinflection in two aspects: First, evaluation might differ slightly, because, in the case of reinflection, the generation of forms that are rare and hard to infer could be not required. In contrast, when performing paradigm completion, all missing forms have to be produced. Second, since multiple forms of the paradigm can be given as input, this can be seen as a multi-source setting. In particular, it enables the application of systems that work on multi-source input. To understand the importance of this, consider the different cases shown in Figure

[1]The definitions of the terms *paradigm completion*, *morphological inflection* and *morphological reinflection* vary slightly throughout the literature, including the work presented in later chapters of this thesis. In this introduction, we will use the definitions outlined here. In each of the following chapters, the terms are defined as applies.

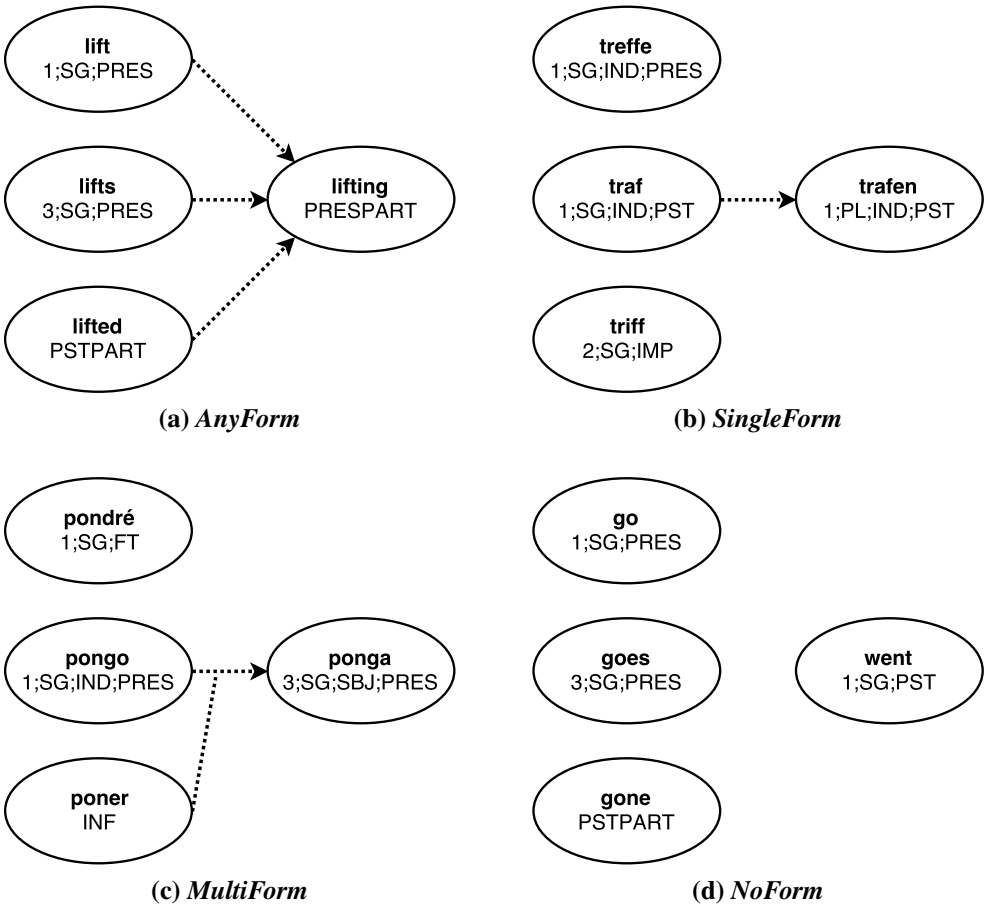


Figure 1.1 – *Different multi-source input configurations for paradigm completion.*

1.1: ANYFORM is the case where one can predict the target form from any of the given source forms, cf. Figure 1.1a. SINGLEFORM is the case where only one form can be used to regularly predict the target form. In the example in Figure 1.1b, it is easy to predict the target form *trafen* from *traf*, but not necessarily from the other two sources. MULTIFORM is the case where multiple forms are *necessary* to predict the target form. For the paradigm of *poner*, which is shown in Figure 1.1c, a combination of two source forms, e.g., *poner* and *pongo*, is needed to infer the target form *ponga*. Finally, in the case of NOFORM, it is impossible to regularly derive the target form from any of the source forms, cf. Figure 1.1d. Leveraging additional source forms is likely to improve performance for the configurations ANYFORM, SINGLEFORM and MULTIFORM, but not for the configuration NOFORM. In particular, as we will show in later chapters, there are cases where it can be beneficial to have information about more than one single

inflected form of a paradigm.

Formalization of the Tasks

We will now describe the three aforementioned tasks in a formal way, developing our notation.

Let $\mathcal{T}$ be the set of morphological tags being expressed in a language and w a lemma in the same language. We define the morphological paradigm π belonging to w as follows:

$$\pi(w) = \Big\{ \big(f_k[w], t_k \big) \Big\}_{k \in \mathcal{T}(w)} \tag{1.1}$$

$f_k[w]$ denotes an inflected form corresponding to tag t_k, and w and $f_k[w]$ are strings formed by letters from an alphabet Σ. Note that, even though we follow the convention to describe word forms as functions of the lemma, in the huge majority of the cases, each inflection is uniquely defined given any other inflected form of the same paradigm and the two respective tags.

Paradigm completion. Given a partial paradigm $\pi(w)_{pt}$ with $\pi(w)_{pt} \subseteq \pi(w)$, the goal of paradigm completion is to produce all inflected forms $f_i[w]$ with $(f_i[w], t_i) \notin \pi(w)_{pt}$. The corresponding tags are supposed to be known.

For example, consider the following partial paradigm, which is a subset of the German paradigm shown in Table 1.1:

$$\pi_{pt}(\text{Schneemann}) = \Big\{ \big(\text{Schneemännern}, \text{PL;DAT} \big) \Big\}$$

We then expect a paradigm completion system to produce all unknown inflected forms, which correspond to the tags SG;NOM, SG;GEN, SG;DAT, SG;ACC, PL;NOM, PL;GEN and PL;ACC.

Morphological reinflection. The task of morphological reinflection consists of predicting a missing form $f_i[w]$ from a paradigm, given another form $f_j[w]$ or w as well as the tag of the target form t_i, and optionally the source tag t_j.

An example from the paradigm in Table 1.1 would be

$$\big(\text{Schneemannes}, \text{SG;GEN}, \text{SG;DAT}\big) \rightarrow \text{Schneemann}$$

SG;GEN and SG;DAT are the source tag and target tag, respectively.

Morphological inflection. The special case of morphological inflection consists of predicting a missing form $f_i[w]$ from a paradigm, given the lemma w together with the target tag t_i. Since the source form is defined to be the lemma, no source tag is needed.

An example would be:

$$\big(\text{Schneemann}, \text{SG;GEN}\big) \rightarrow \text{Schneemannes}$$

1.2.3 Morphological Analysis

The morphological analysis task called **morphological segmentation** aims to either divide a word into morphemes, i.e., its smallest meaning-bearing units, or into morphs, the surface forms thereof. Traditionally, most research described in the NLP literature has focused on *surface segmentation*, whereby a word w is segmented into a sequence of substrings whose concatenation results in the entire word; see Ruokolainen et al. (2016) for a survey. In contrast, we are interested in *canonical segmentation*, where w is divided into a sequence of standardized segments. In order to understand the difference, consider the following example: the surface segmentation of the complex English word *achievability* is *achiev+abil+ity*, whereas its canonical segmentation is *achieve+able+ity*. In particular, canonical segmentation includes restoring the alterations made during word formation. This version of the task has several representational advantages over surface segmentation, e.g., whether two words share a morpheme is no longer obfuscated by orthography. However, it also introduces a more complicated algorithmic challenge: besides segmenting a word, one must also reverse orthographic changes, e.g., perform the mapping *achievability*$\mapsto$*achieveableity*. Canonical versions of morphological segmentation have been introduced multiple times in the NLP literature (Kay, 1977; Naradowsky and Goldwater, 2009; Cotterell et al., 2016b).

In general, segmentations are useful in a diverse set of applications, e.g., automatic speech recognition (Afify et al., 2006), keyword spotting (Narasimhan et al., 2014), machine translation (Clifton and Sarkar, 2011) and parsing (Seeker and Çetinoğlu, 2015).

Formalization of the Task

We cast segmentation as a sequence-to-sequence transduction task. Given an alphabet Σ (e.g., the 26 letters of the English alphabet), the goal of canonical segmentation is to map a word $w \in \Sigma^*$ (e.g., w=*achievability*) to a sequence $c \in \Omega^*$ of characters, representing its correct segmentation and marking the borders of the sub-word units it is formed of (e.g., c=*achieve+able+ity*). We define $\Omega = \Sigma \cup \{+\}$, where $+$ is a distinguished separation symbol. Additionally, we can write the segmented form as $c = \sigma_1 + \sigma_2 + \ldots + \sigma_n$, where each segment $\sigma_i \in \Sigma^*$ and n is the number of canonical segments.

1.3 Neural Networks

Artificial neural networks are strong computational models suitable for a broad range of applications inside and outside of NLP. Their name dates back to attempts

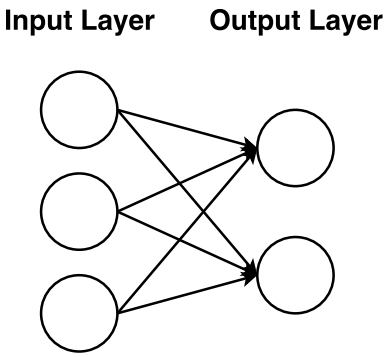


Figure 1.2 – *A simple perceptron without any hidden layers.*

to find mathematical representations of information processing in biological systems (McCulloch and Pitts, 1943; Widrow and Hoff, 1960), even though some of them arguably lack biological plausibility. In particular, artificial neural networks are much smaller than the brain of a mammal; an artificial network might have thousands of processor units, whereas a mammalian brain has billions of neurons.

The main neural network architecture we use and extend in our work is an attention-based encoder-decoder RNN as presented first by Bahdanau et al. (2015) for neural machine translation. Though there exist a multitude of different neural network types, we will limit our descriptions in this introduction to those needed to understand this thesis.

1.3.1 Perceptrons

The most basic neural network model is a perceptron. It consists of an input layer, i.e., vectors representing the input, a variable number of hidden layers and an output layer. Except for the input layer which does not perform any computations, every layer is realized by a weight matrix followed by a non-linearity. The number of dimensions of the vectors representing the results of each layer is called the number of units per layer. There are two common ways of counting the number of layers of a perceptron, which are to include or not the input layer. We adopt the view that, since it does not compute anything, the input layer should not be counted, such that we call an architecture without hidden layers a single-layer perceptron, cf. Figure 1.2. In contrast, a perceptron with one or more hidden layers is called a multi-layer perceptron (MLP), cf. Figure 1.3.

In perceptrons, connections from each layer exclusively lead to subsequent layers. In particular, the output of a layer is not used for calculations in earlier or the same layers. Therefore, a perceptron belongs to the class of feedforward networks. In contrast, network architectures which make use of recurrent connections are called recurrent neural networks.

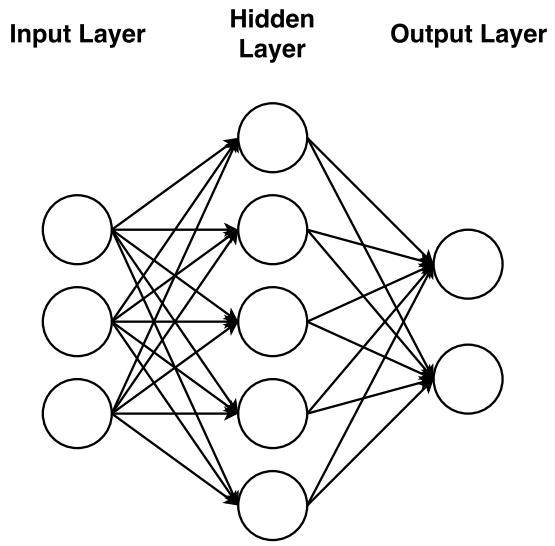


Figure 1.3 – *An MLP with one hidden layer.*

Formally, every layer of an MLP is a function $f(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^c$ that produces a c-dimensional output for a d-dimensional input $\mathbf{x}$ according to the following formula:

$$f(\mathbf{x}) = g(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{1.2}$$

In a hidden layer, g can be any non-linear function, typically the *sigmoid* or the *hyperbolic tangent* function. In the output layer, g is usually realized as the *softmax* function. $\mathbf{W}$ is a weight matrix and $\mathbf{b}$ a bias vector. Both of them are usually learned from the available training data, using backpropagation. During this process, the parameters of the network are updated according to the error made on the training examples.

Perceptrons get more expressive with increasing depth, i.e., a larger number of hidden layers. In particular, a single-layer perceptron can only solve linearly separable problems (Hertz et al., 1991).

1.3.2 Recurrent Neural Networks

Because the numbers of units in their input and output layers are defined beforehand, feedforward networks like perceptrons are limited to inputs of a fixed size. Correspondingly, they always produce equally fixed-sized outputs. This limitation can be relaxed by padding, i.e., by filling empty dimensions with placeholders, e.g., zeros. However, even then the maximum input and output sizes still need to be chosen beforehand. In contrast, recurrent neural networks (RNNs) (Elman,

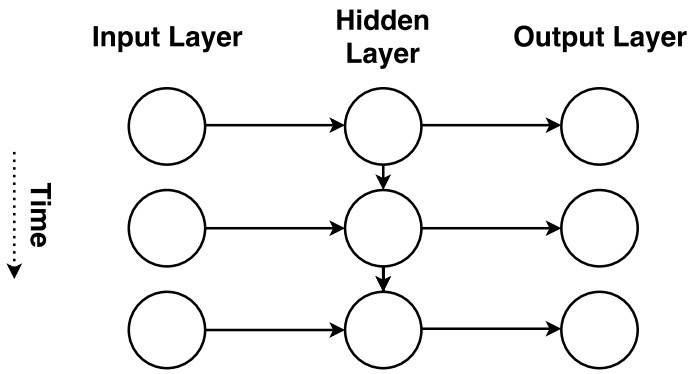


Figure 1.4 – *A simple RNN architecture, expanded in the dimension of time.*

1990) were developed to handle input sequences of variable length without the need for padding. They treat the input as a sequence of separate units and obtain a representation by accumulating information going through the input one by one.

An RNN computes hidden representations for each element in a given input, which is represented by a sequence of embedding vectors $x = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_T})$ of length T, as follows:

$$\mathbf{h_t} = g(\mathbf{U^h x_t} + \mathbf{W^h h_{t-1}} + \mathbf{b^h}) \tag{1.3}$$

Again, g is a non-linear function. An overview of a simple RNN architecture is shown in Figure 1.4. $\mathbf{W^h}$ and $\mathbf{U^h}$ are weight matrices and $\mathbf{b^h}$ is a bias vector.

One of the main drawbacks of RNNs is the vanishing gradient problem, which is a general problem for deep neural networks, i.e., networks with multiple layers: During backpropagation, parameters are updated proportionally to the gradient of the error function. However, this gradient is computed according to the chain rule, which leads to vanishingly small gradients for high numbers of layers. The vanishing gradient problem affects RNNs because they can be unfolded in time, resulting in an architecture with multiple hidden layers. Thus, for longer inputs, the weights are kept from being updated in a meaningful way, causing the network's performance to decrease rapidly.

In order to overcome this, extensions of the original RNN architecture have been proposed: **long-short term memory networks** (LSTMs) (Hochreiter and Schmidhuber, 1997) and **gated recurrent units** (GRUs) (Cho et al., 2014b). Both share the idea of controlling the information at each time step by giving the network a mechanism to keep more or less of the past history as suitable. Performances of the two types of extensions are typically similar. However, GRUs—which we employ in our models—are slightly simpler, since they contain fewer parameters (Cho et al., 2014b).

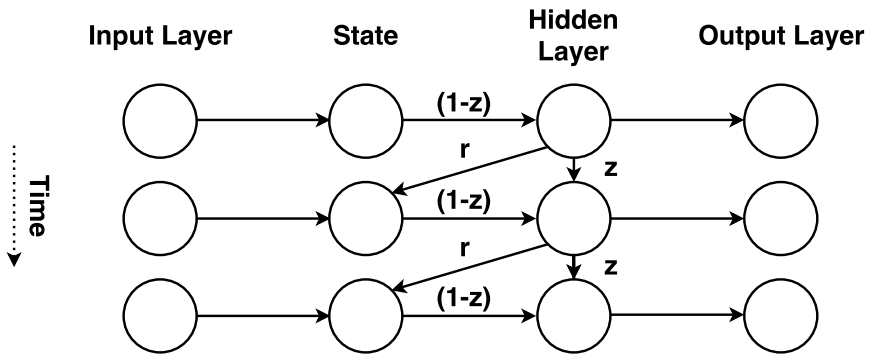


Figure 1.5 – *The architecture of a GRU, expanded in the dimension of time.*

The formulas defining a GRU with input $x = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_T})$ are:

$$\mathbf{z_t} = \sigma(\mathbf{U^z x_t} + \mathbf{W^z h_{t-1}} + \mathbf{b^z}) \tag{1.4}$$
$$\mathbf{r_t} = \sigma(\mathbf{U^r x_t} + \mathbf{W^r h_{t-1}} + \mathbf{b^r}) \tag{1.5}$$
$$\mathbf{s_t} = \tanh(\mathbf{U^s x_t} + \mathbf{W^s}(\mathbf{h_{t-1}} * \mathbf{r_t}) + \mathbf{b^s}) \tag{1.6}$$
$$\mathbf{h_t} = (\mathbf{1} - \mathbf{z_t}) * \mathbf{s_t} + \mathbf{z_t} * \mathbf{h_{t-1}} \tag{1.7}$$

The h-dimensional vectors $\mathbf{s_t}$ and $\mathbf{h_t}$ are called the state and the hidden state of the RNN, respectively. $\mathbf{z_t}$ and $\mathbf{r_t}$ are the so-called gates of the GRU. They control the information flow through the network by application of the *sigmoid* function to a weighted sum of the input $\mathbf{x_t}$ and the preceding hidden state $\mathbf{h_{t-1}}$. $\mathbf{U^j} \in \mathbb{R}^{h \times d}$ and $\mathbf{W^j} \in \mathbb{R}^{h \times h}$ are weight matrices, and the $\mathbf{b^j}$ are bias vectors, $j \in \{r, s, z\}$. Weights and biases are parameters to be learned during training. The architecture of a GRU is displayed in Figure 1.5.

The corresponding formulas for an LSTM, as shown in Figure 1.6, are:

$$\mathbf{i_t} = \sigma(\mathbf{U^i x_t} + \mathbf{W^i h_{t-1}} + \mathbf{b^i}) \tag{1.8}$$
$$\mathbf{f_t} = \sigma(\mathbf{U^f x_t} + \mathbf{W^f h_{t-1}} + \mathbf{b^f}) \tag{1.9}$$
$$\mathbf{o_t} = \sigma(\mathbf{U^o x_t} + \mathbf{W^o h_{t-1}} + \mathbf{b^o}) \tag{1.10}$$
$$\mathbf{q_t} = \tanh(\mathbf{U^q x_t} + \mathbf{W^q h_{t-1}} + \mathbf{b^q}) \tag{1.11}$$
$$\mathbf{p_t} = \mathbf{f_t} * \mathbf{p_{t-1}} + \mathbf{i_t} * \mathbf{q_t} \tag{1.12}$$
$$\mathbf{h_t} = \mathbf{o_t} * \tanh(\mathbf{p_t}) \tag{1.13}$$

As for the GRU, $\mathbf{h_t}$ is the hidden state of the network at time step t, and $\mathbf{U^j} \in \mathbb{R}^{h \times d}$, $\mathbf{W^j} \in \mathbb{R}^{h \times h}$ and $\mathbf{b^j}$ for $j \in \{i, f, o, q\}$ are again weight matrices and bias vectors. An LSTM has three gates: the input gate $\mathbf{i_t}$, the forget gate $\mathbf{f_t}$ and the output gate $\mathbf{o_t}$. All gates are put into practice as non-linear functions applied to weighted combinations of the input vector $\mathbf{x_t}$ and the last hidden state $\mathbf{h_{t-1}}$.

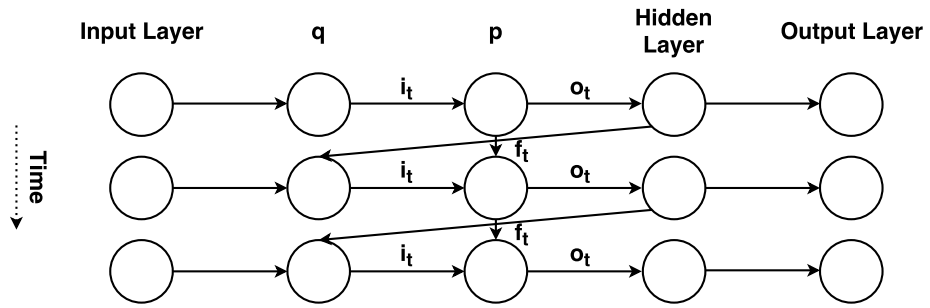


Figure 1.6 – *The architecture of an LSTM, expanded in the dimension of time.*

RNNs are mainly used for two categories of tasks: sequence labeling and classification. For the former, each hidden state is fed into an output layer, producing one output for each time step. For the latter, the goal is typically to produce one output for an entire input sequence. This makes it necessary to decouple the output length from the input length. Thus, for classification, it is common to use the last hidden state $\mathbf{h_T}$, which is supposed to represent the entire input, as input to a final output layer.

1.3.3 Encoder-Decoder Recurrent Neural Networks

Cho et al. (2014b) and Sutskever et al. (2014) independently developed neural sequence-to-sequence models based on RNNs, the so-called encoder-decoder networks or encoder-decoder RNNs. In contrast to earlier applications of RNNs, where either the last hidden state had been used for classification or one output had been produced for each element in the input sequence, the new architecture made it possible to produce variable-length output for variable-length input. Such variable-length sequence-to-sequence tasks are something commonly found when working with natural language, e.g., for translating an input sentence into an output sentence or for question answering, where the question and text with additional information form the input sequence and the answer is the output sequence. Thus, encoder-decoder RNNs are extremely suitable for NLP applications.

The basic sequence-to-sequence model by Cho et al. (2014a) and Sutskever et al. (2014) consists of two parts: an **encoder** which processes the input, and a **decoder** which generates the output. The encoder is implemented as an RNN that reads the vectors of the input sequence $x = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_{T_x}})$ and encodes them into a fixed-length context vector $\mathbf{c}$, computing the hidden states $\mathbf{h_t}$ and $\mathbf{c}$ as:

$$\mathbf{h_t} = f(\mathbf{x_t}, \mathbf{h_{t-1}}) \tag{1.14}$$

$$\tag{1.15}$$

f is again a non-linear function, e.g., the *sigmoid* function.

The decoder, implemented as a second RNN, produces the output $y = (y_1, y_2, \ldots, y_{T_y})$. Note that the length of the output T_y is not constrained to be equal to the input length T_x. The decoder is trained to predict each element of the output y_t dependent on $\mathbf{c}$ and all previous predictions $y_1, \ldots, y_{t-1}$:

$$p(y) = \prod_{t=1}^{T_y} p(y_t | \{y_1, \ldots, y_{t-1}\}, \mathbf{c}) \tag{1.16}$$

with each conditional probability being modeled with an RNN as:

$$p(y_t | \{y_1, \ldots, y_{t-1}\}, \mathbf{c}) = g(y_{t-1}, \mathbf{s_t}, \mathbf{c}) \tag{1.17}$$

Again, g is a non-linear function and $\mathbf{s_t}$ is the hidden state of the decoder RNN. $\mathbf{c}$ is a so-called context vector; most commonly, the last hidden state of the encoder $\mathbf{h_{T_x}}$ is used.

Even though for the work presented in this thesis we only employ RNN encoder-decoder models, sequence-to-sequence architectures are not limited to recurrent networks. There exists work on encoder-decoder models based on convolutional neural networks (CNNs) (Kalchbrenner et al., 2016; Gehring et al., 2017) or even simple feedforward networks with attention-mechanisms for sequence-to-sequence transduction (Vaswani et al., 2017). Those are reported to be computationally faster to train, because they can be parallelized. However, due to them being invented only recently, we did not try their performance on our tasks and will leave this for future work.

1.3.4 Attention Mechanism

A weakness limiting the performance of the basic encoder–decoder approach is that the network needs to be able to compress all necessary information about the entire input sequence into a single fixed-length vector. Since this is complicated for longer input sequences, Cho et al. (2014a) showed that the performance decreases quickly when the length of the input grows.

In order to overcome this issue, Bahdanau et al. (2015) introduced a so-called attention mechanism. The main idea of their approach was to get rid of the fixed-length context vector $\mathbf{c}$. Instead, they proposed to calculate a new context vector for each output element as a weighted combination of all hidden states of the encoder. In this way, the information does not have to be compressed into one single vector anymore.

Formula 1.16, which describes the decoder, thus changes as follows:

$$p(y) = \prod_{t=1}^{T_y} g(y_{t-1}, \mathbf{s_t}, \mathbf{c_t}) \tag{1.18}$$

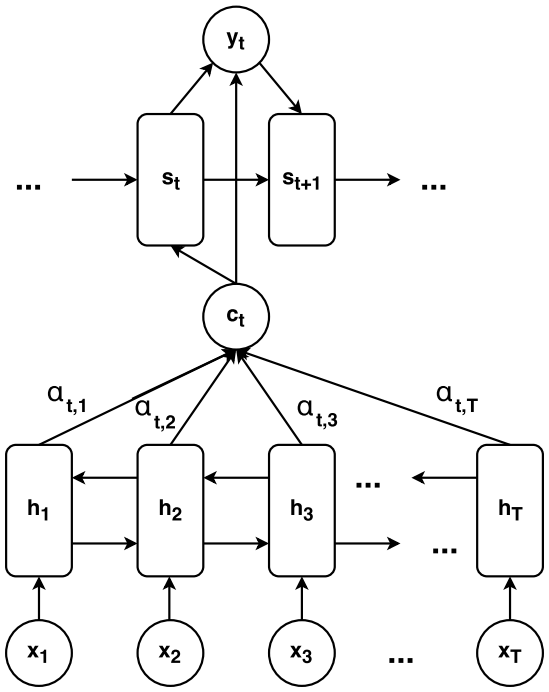


Figure 1.7 – *Schematic overview of the attention-based encoder-decoder model.*

All formerly introduced variables denote the same as before. The single fixed-length context vector $\mathbf{c}$ has been substituted by a time step-dependent context vector $\mathbf{c_t}$, being the weighted sum of the hidden states $(\mathbf{h_1}, \ldots, \mathbf{h_{T_x}})$ produced by the encoder. It is calculated using the following formulas:

$$e_{ij} = a(\mathbf{s_{i-1}}, \mathbf{h_j}) \tag{1.19}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{1.20}$$

$$\mathbf{c_i} = \sum_{j=1}^{T_x} \alpha_{ij} \mathbf{h_j} \tag{1.21}$$

The weights α_{ij} are called attention weights and are functions of the energies e_{ij}. Bahdanau et al. (2015) suggested to parameterize a as a feedforward neural network and to train it jointly with the other components. More specifically, they proposed to use a multilayer perceptron in order to keep the computational cost relatively low, as this needs to be evaluated $T_x T_y$ times for an input of length T_x and an output of length T_y. Thus, the formula to calculate a is:

$$a(\mathbf{s_{i-1}}, \mathbf{h_j}) = (\mathbf{v^a})^T \tanh(\mathbf{W^a s_{i-1}} + \mathbf{U^a h_j} + \mathbf{b^a}) \tag{1.22}$$

$\mathbf{W^a}$ and $\mathbf{U^a}$ are weight matrices. $\mathbf{v^a}$ and $\mathbf{b^a}$ are vectors.

Additionally to the attention mechanism, the model of Bahdanau et al. (2015) which we build on in this work employs a bidirectional encoder, with the final encoder hidden states being the concatenation of the hidden states of a forward and a backward RNN. The resulting attention-based encoder-decoder architecture is shown in Figure 1.7.

1.4 Related Work

Now we will describe how our morphological generation and analysis tasks as well as similar ones have been approached in the past. Subsequently, we will give a general overview of other NLP tasks which can be tackled with neural sequence-to-sequence models.

1.4.1 Morphology

Paradigm completion and morphological reinflection. In the last two years, a lot of important work on the tasks of paradigm completion and morphological reinflection has been done in the context of the SIGMORPHON 2016 shared task on morphological reinflection (Cotterell et al., 2016a) and its follow-up edition, the CoNLL-SIGMORPHON 2017 shared task on universal morphological reinflection (Cotterell et al., 2017a).

In 2016, the shared task was limited to morphological inflection (task 1) and morphological reinflection (tasks 2 and 3). The submitted systems could roughly be sorted into 3 categories. The first category consisted of neural sequence-to-sequence models and contained the strongest systems in the competition. Those were namely RNN encoder-decoder models with soft (Kann and Schütze, 2016) or hard attention (Aharoni et al., 2016), as well as one encoder-decoder model that employed a convolutional layer over the character input (Östling, 2016).

The systems of the second category were built upon earlier work by Durrett and DeNero (2013). They first computed an unsupervised alignment for the pairs of source and target forms in the training set. This alignment was then used to extract edit operations that transformed the input into the target string. Finally, a system was trained on predicting the right transformations to obtain the output from a combination of the input string and the morphological tags. Alegria and Etxeberria (2016) and Nicolai et al. (2016) employed weighted finite state transducers for those applications of edit operations. In turn, Liu and Mao (2016) and King (2016) used a semi-Markov conditional random field (CRF) (Sarawagi and Cohen, 2005). Nicolai et al. (2016) further introduced a reranker to rescore the answer candidates obtained by their system.

Finally, the last category covered all systems that made use of linguistically inspired heuristics. They effectively reduced the problem to multi-class classification. Taji et al. (2016) took ideas from Eskander et al. (2013) which had originally been intended for the creation of a morphological analyzer and generator, using data that had been manually segmented and clustered by lexeme. Sorokin (2016) followed ideas by Ahlberg et al. (2014, 2015).

The 2017 edition of the shared task was concerned with morphological inflection (task 1) and paradigm completion (task 2). Due to the success of encoder-decoder networks in 2016, most systems developed for 2017 were neural sequence-to-sequence models. In particular, most teams used some version of an RNN, e.g., (Makarov et al., 2017; Bergmanis et al., 2017; Zhou and Neubig, 2017).

Finally, besides systems designed for the two shared tasks and the work presented in this thesis, Cotterell et al. (2017b) considered a multi-source setting for paradigm completion. They modeled paradigms using graphical models with neural parameterizations, defined over multiple string-valued random variables. Thus, they could jointly decode entire paradigms, leveraging all available source forms. Additionally, Zhou and Neubig (2017) presented multi-space variational encoder-decoders for the task of morphological reinflection.

Earlier influential work on paradigm completion or reinflection—both neural and non-neural—which is not mentioned above, included but was by no means limited to (Dreyer et al., 2008; Hulden et al., 2014; Nicolai et al., 2015; Faruqui et al., 2016b).

Morphological segmentation. Several approaches to morphological segmentation have been proposed in the literature. In the unsupervised realm, most work was based on the principle of minimum description length (Rissanen, 1989). Goldsmith (2001) and, besides other follow-up work, Lee and Goldsmith (2016) introduced LINGUISTICA for the unsupervised learning of segmentation of European languages. In particular, the program took unlabeled corpora of varying sizes as input and produced partial morphological analyses of the words, splitting them into morphemes and categorizing them. Goldsmith (2001) developed a set of heuristics to build a probabilistic morphological grammar. Minimum description length analysis was used to decide if a modification proposed by those heuristics should be applied or discarded.

Creutz and Lagus (2002) presented two different methods for the unsupervised splitting of words into morpheme-like units. The first was also based on the minimum description length principle to simultaneously measure the goodness of the representation and the model complexity, whereas the second made use of maximum likelihood optimization. Their approach, instead of expecting a single one for each of prefix, stem and suffix, was especially suitable for languages

with agglutinative morphology, where words can consist of long sequences of segments. Based on the ideas described by Creutz and Lagus (2002), another program for unsupervised morphological segmentation called MORFESSOR was developed (Creutz et al., 2007). As using the corpus vocabulary instead of the corpus led to segmentation results closer to linguistic morphemes, MORFESSOR was trained on a collection of word types instead of word tokens. Words below a certain frequency threshold were excluded from training.

Later on, Kohonen et al. (2010) extended MORFESSOR to a semi-supervised version by adding known linguistic segmentations into the data likelihood function and performing optimization of separate weights for unlabeled and labeled data. They showed that performance improved quickly even for only few annotated examples, highlighting the potential benefits of minimal supervision when sets of labeled data are available, but not large enough to enable supervised approaches to work well.

Supervised approaches to morphological segmentation exist as well. Most notably, Ruokolainen et al. (2013) developed a supervised approach for the segmentation task based on CRFs, which they subsequently extended to also work in a semi-supervised way (Ruokolainen et al., 2014) using letter successor variety features (Hafer and Weiss, 1974). Further, Cotterell et al. (2015) improved performance with a semi-Markov CRF. Finally, Wang et al. (2016) achieved state-of-the-art results on surface morphological segmentation using a window LSTM.

Several groups presented work on canonical segmentation, calling the task a set of different names, e.g., Kay (1977); Naradowsky and Goldwater (2009); Cotterell et al. (2016b).

Other morphological tasks. Research on computational morphology has produced a large variety of tasks and respective approaches. While some methods have been carefully designed by hand (Koskenniemi, 1983; Buckwalter, 2004), recently, machine learning methods including neural networks have been gaining popularity. Those can be supervised—like most approaches presented in this thesis—, semi-supervised or unsupervised.

Of special interest here are the tasks of morphological tagging (Müller et al., 2013) and morphological analysis (Schmid et al., 2004; Nicolai and Kondrak, 2017), i.e., producing morphological tags for a given word with or without context, since they can be seen as the reverse of paradigm completion. Also related to the morphological generation tasks in this thesis is work concerned with the automatic creation of morphological lexicons (Eskander et al., 2013; Faruqui et al., 2016a). Even though the way to approach this is usually unrelated, the final products are quite similar. Note in particular that a full-coverage lexicon can be used directly for morphological reinflection and would be the product of paradigm

completion for all lemmas of a language.

Other tasks in the realm of morphology include compound splitting (Koehn and Knight, 2003; Macherey et al., 2011), prediction of derivational forms (Vylomova et al., 2017), or—even though it is not morphology in the stricter sense—portmanteau creation (Deri and Knight, 2015).

Recently, the ability of machine learning models to deal with morphology has been investigated explicitly by several groups. Belinkov et al. (2017) performed several experiments in order to answer the question what neural machine translation models learn about morphology. They found that character-based models were better at learning morphology than word-based ones. Further, lower network layers seemed to be mainly responsible for handling morphology, while the attentional decoder did not learn much about it. Avraham and Goldberg (2017) focused on the relationship of semantics and morphology in word embeddings. Their main result was that including morphological information into word embeddings could harm performance on semantic tasks. Due to this trade-off, they concluded that, unless needed for the task at hand, it would be better to ignore morphology during the creation of word embeddings.

1.4.2 Sequence-to-Sequence Models in NLP

Morphological generation and analysis are by far not the only tasks for which neural sequence-to-sequence models constitute the state of the art. Even though those models had originally been designed for translation (Sutskever et al., 2014; Cho et al., 2014a; Bahdanau et al., 2015), they have shown to be useful for a wide and diverse area of applications. As of today, those architectures have been successfully applied to many sequence-to-sequence tasks in NLP, e.g., syntactic parsing (Vinyals et al., 2015), automatic speech recognition (Graves and Schmidhuber, 2005; Graves et al., 2013) or language correction (Xie et al., 2016).

Attention for sequence-to-sequence models was also studied extensively. So-called hard attention (Aharoni and Goldberg, 2017) was introduced in contrast to Bahdanau et al. (2015)'s soft attention. Thereby, the model attends to one single input state at a time and either adds a symbol to the output sequence or moves the position of the attention pointer to the next hidden state of the encoder. Aharoni and Goldberg (2017) argued that this was especially suitable for tasks that exhibit an almost monotonic alignment between the input and output sequences, as it is the case for paradigm completion or morphological segmentation. Findings of the CoNLL-SIGMORPHON 2017 shared task (Cotterell et al., 2017a) indicated that hard attention might be of particular interest for extreme low-resource scenarios. Also relevant for the multi-source setting of the paradigm completion task is work like the one by Libovický and Helcl (2017): The authors proposed two novel approaches to combine the outputs of attention mechanisms over each source for

multi-source sequence-to-sequence tasks, which they called flat and hierarchical attention. They argued that their attention strategies were interpretable and took into account the roles of the individual source sequences explicitly. Thus, they might be something that could easily and potentially beneficially be combined with our work.

Relevant for multi-source paradigm completion are also multi-source sequence-to-sequence approaches for machine translation: Zoph and Knight (2016) made simultaneous use of source sentences in multiple languages with the goal of finding the best match possible in the target language. They applied transformations to the hidden states of the encoders that were the input to the decoder. Firat et al. (2016) presented an approach that translated from any of N source languages to any of M target languages, using encoders and decoders specific to each of them. One common attention mechanism was shared between all languages.

1.5 Proposed Approaches

RNN encoder-decoders models are strong computational models, which define the state of the art for a variety of different NLP tasks. Their main drawback is that they require large amounts of training data. In this thesis, we present multiple ways to mitigate this problem and to make neural sequence-to-sequence models applicable to morphological tasks even in low-resource settings.

The basis of this thesis is the presentation of the morphological encoder-decoder model MED, a language-independent character-level sequence-to-sequence model for morphological reinflection or paradigm completion, which we extend in later chapters. In contrast to earlier work by Faruqui et al. (2016b), we train one single model on all mappings from source to target forms that appear in the training set of a given language. This way, we perform multi-task training on all subtasks, effectively sharing all of the network's parameters. This radically reduces the amount of training data needed, since most morphological patterns occur in many source-target tag pairs. The key enabler for our approach is a novel representation we use for the task. We encode each input as a continuous sequence of the morphological tags of the source form, the morphological tags of the target form and the characters of the source form. The output is the sequence of letters the target form is composed of. We show that MED defines a new state of the art for the SIGMORPHON 2016 shared task dataset and, together with the right data augmentation techniques, also performs on par with the winning system of task 1 of the CoNLL-SIGMORPHON 2017 shared task in all but the lowest-resource track.

We further investigate methods to deal with morphology in extreme low resource settings. One possible approach we present consists of extending MED to

enable cross-lingual transfer by adding an additional input: a tag representing the language of the sample at hand, similar to an idea introduced by Johnson et al. (2017) for machine translation. By doing so, we can train a single neural network model on multiple languages, and thus exploit annotations in a high-resource language to train a system for an extremely low-resource language. We analyze the effectiveness of our method for morphological reinflection on multiple language pairs belonging to different language families. We further address the research question which languages might be especially apt for this type of morphological transfer. Our results hint that the degree of performance gain depends on the relatedness of the high-resource and the low-resource language.

Another approach is to extend MED to enable semi-supervised training. This is based on the idea that an abundance of unlabeled data can safely be assumed to be available for each language in the focus of NLP. We obtain a semi-supervised model by treating unlabeled words as training examples for an autoencoding (Vincent et al., 2010) task and performing multi-task training. We expect this to be beneficial for the following reasons: (i) The decoder's character language model can be trained using unlabeled data. (ii) Training on a second task mitigates the overfitting problem. (iii) By forcing the model to additionally learn to autoencode, we give it a strong prior to copy the input string. This might be advantageous, because often many forms of a paradigm share the same stem, e.g., ***smil****ing* and ***smil****ed*. We further present a second way of semi-supervised training: using autoencoding of random strings as an auxiliary task. We find that for our experimental settings and non-templatic languages the performance gain is comparable to using corpus words.

Turning to paradigm completion, we finally address how to make use of information emerging from the general structure of paradigms. We consider two questions: How can we leverage information from a set of given entire paradigms? Further, how can we make use of multiple input sources for completing a certain paradigm, and how does the usefulness of additional forms depend on properties of the language and the paradigm?

As an intended answer to the first question, we present CIS, an algorithm selecting the most suitable source form for a target form, given the set of all available source forms. Addressing the second question, we propose two approaches: First, we experiment with a multi-source input version of MED, leaving the selection of the best source for each target form to the model. Second, we introduce a novel alternative way to make use of additional given forms: *paradigm adaptation*, i.e., fine-tuning the encoder-decoder model on available source forms before generating the target forms. The motivation for paradigm adaptation is that, since members of the same paradigm often consist of similar character sequences, we expect each available input form to contain valuable information about what the unknown target form looks like. We assume that we can pass this information to

the generation part of the model by fine-tuning it on new examples created from the partial paradigm.

Finally, we apply the same neural sequence-to-sequence architecture to morphological segmentation and show that encoder-decoder models are able to improve over the previous state of the art also for this analysis task.

Overall, we hope that this thesis will make a contribution to research on morphology generation and analysis, and help to make neural network models applicable in low-resource settings.

Chapter 2

Single-Model Encoder-Decoder with Explicit Morphological Representation for Reinflection

Single-Model Encoder-Decoder with Explicit Morphological Representation for Reinflection

Katharina Kann and Hinrich Schütze
Center for Information & Language Processing
LMU Munich, Germany
kann@cis.lmu.de

Abstract

Morphological reinflection is the task of generating a target form given a source form, a source tag and a target tag. We propose a new way of modeling this task with neural encoder-decoder models. Our approach reduces the amount of required training data for this architecture and achieves state-of-the-art results, making encoder-decoder models applicable to morphological reinflection even for low-resource languages. We further present a new automatic correction method for the outputs based on edit trees.

1 Introduction

Morphological analysis and generation of previously unseen word forms is a fundamental problem in many areas of natural language processing (NLP). Its accuracy is crucial for the success of downstream tasks like machine translation and question answering. Accordingly, learning morphological inflection patterns from labeled data is an important challenge.

The task of morphological reinflection (MRI) consists of producing an inflected form for a given source form, source tag and target tag. A special case is morphological inflection (MI), the task of finding an inflected form for a given lemma and target tag. An English example is "tree"+PLURAL → "trees". Prior work on MI and MRI includes machine learning models and models that exploit the paradigm structure of the language (Ahlberg et al., 2015; Dreyer, 2011; Nicolai et al., 2015).

In this work, we propose the neural encoder-decoder MED – Morphological Encoder-Decoder – a character-level sequence-to-sequence attention model that is a language-independent solution for MRI. In contrast to prior work, we train a *single* model that is trained on all source to target mappings of the language that are attested in the training set. This radically reduces the amount of training data needed for the encoder-decoder because most MRI patterns occur in many source-target tag pairs. In our model design, what is learned for one pair can be transferred to others.

The key enabler for this single-model approach is a novel representation we use for MRI. We encode the input as a single sequence of (i) the morphological tags of the source form, (ii) the morphological tags of the target form and (iii) the sequence of letters of the source form. The output is the sequence of letters of the target form. As the decoder produces each letter, the attention mechanism can focus on *the input letter sequence* for parts of the output that simply copy the input. For other parts of the output, e.g., an inflectional ending that is predicted using the target tags, the attention mechanism can focus on *the target morphological tags*. In more complex cases, *simultaneous attention can be paid to subsequences of all three input types* – source tags, target tags and input letter sequence. We can train a single generic encoder-decoder per language on this representation that can handle all tag pairs, thus making it possible to make efficient use of the available training data. MED outperformed other systems on the SIGMORPHON16 shared task[1] for all ten languages that were covered (Kann and Schütze, 2016; Cotterell et al.,).

We also present POET – Prefer Observed Edit Trees – a new generic method for correcting the output of an MRI system. The combination of MED and POET is state-of-the-art or close to it on a CELEX-based evaluation of MRI even though this evaluation makes it difficult to exploit gener-

[1] ryancotterell.github.io/sigmorphon2016/

alizations across tag pairs.

2 Model Description

Neural network model. Our model is based on the network architecture proposed by Bahdanau et al. (2014) for machine translation.[2] They describe the model in detail; unless we explicitly say so in the description of our model below, we use the same network configuration as Bahdanau et al. (2014).

Bahdanau et al. (2014)'s model is an extension of the recurrent neural network (RNN) encoder-decoder developed by Cho et al. (2014) and Sutskever et al. (2014). The encoder of the latter consists of an RNN that reads an input sequence of vectors x and encodes it into a fixed-length context vector c, computing hidden states h_t and c by

$$h_t = f(x_t, h_{t-1}), \quad c = q(h_1, ..., h_{T_x}) \tag{1}$$

with nonlinear functions f and q. The decoder is trained to predict each output y_t dependent on c and previous predictions y_1, ..., y_{t-1}:

$$p(y) = \prod_{t=1}^{T_y} p(y_t|\{y_1, ..., y_{t-1}\}, c) \tag{2}$$

with $y = (y_1, ..., y_{T_y})$ and each conditional probability being modeled with an RNN as

$$p(y_t|\{y_1, ..., y_{t-1}\}, c) = g(y_{t-1}, s_t, c) \tag{3}$$

where g is a nonlinear function and s_t is the hidden state of the RNN.

Bahdanau et al. (2014) proposed an attention-based extension of this model that allows different vectors c_t for each step by automatic learning of an alignment model. Additionally, they made the encoder bidirectional: each hidden state h_j at time step j does not only depend on the preceding, but also on the following input:

$$h_j = \left[\overrightarrow{h_j^T}; \overleftarrow{h_j^T}\right]^T \tag{4}$$

The formula for $p(y)$ changes as follows:

$$p(y) = \prod_{t=1}^{T_y} p(y_t|\{y_1, ..., y_{t-1}\}, x) \tag{5}$$

$$= g(y_{t-1}, s_t, c_t) \tag{6}$$

[2]Our implementation of MED is based on `github.com/mila-udem/blocks-examples/tree/master/machine_translation`.

with s_t being an RNN hidden state for time t and c_t being the weighted sum of the annotations $(h_1, ..., h_{T_x})$ produced by the encoder, using the attention weights. Further descriptions can be found in (Bahdanau et al., 2014).

The final model is a multilayer network with a single maxout (Goodfellow et al., 2013) hidden layer that computes the conditional probability of each element in the output sequence (a letter in our case, (Pascanu et al., 2014)). As MRI is less complex than machine translation, we reduce the number of hidden units and embedding size. After initial experiments, we fixed the hyperparameters of our system and did not further adapt them to a specific task or language. Encoder and decoder RNNs have 100 hidden units each. For training, we use stochastic gradient descent, Adadelta (Zeiler, 2012) and a minibatch size of 20. We initialize all weights in the encoder, decoder and the embeddings except for the GRU weights in the decoder with the identity matrix as well as all biases with zero (Le et al., 2015). We train all models for 20,000 iterations. We settled on this number in early experimentation because training usually converged before that limit.

MED is an ensemble of five RNN encoder-decoders. The final decision is made by majority voting. In case of a tie, the answer is chosen randomly among the most frequent predictions.

Input and output format. We define the alphabet Σ_{lang} as the set of characters used in the application language. As each morphological tag consists of one or more subtags, e.g. "number" or "case", we further define Σ_{src} and Σ_{trg} as the set of morphological subtags seen during training as part of the source tag and target tag, respectively. Let S_{start} and S_{end} be predefined start and end symbols. Then each input of our system is of the format $S_{\text{start}}\Sigma_{\text{src}}{}^{+}\Sigma_{\text{trg}}{}^{+}\Sigma_{\text{lang}}{}^{+}S_{\text{end}}$. In the same way, we define the output format as $S_{\text{start}}\Sigma_{\text{lang}}{}^{+}S_{\text{end}}$.

A sample input for German is *<w> IN=pos=ADJ IN=case=GEN IN=num=PL OUT=pos=ADJ OUT=case=ACC OUT=num=PL i s o l i e r t e r </w>*. The system should produce the corresponding output *<w> i s o l i e r t e </w>*. The high-level structure of MED can be seen in Figure 1.

POET. We now describe POET (Prefer Observed Edit Trees), a new generic method for correcting the output of an MRI system. We use it in combination with MED in this paper, but it can in

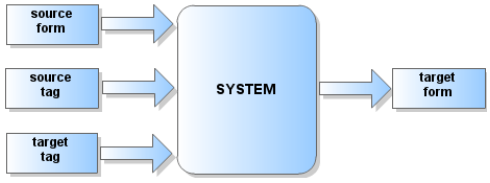


Figure 1: Overview of MED

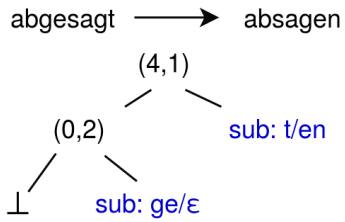


Figure 2: Edit tree for the inflected form *abgesagt* "canceled" and its lemma *absagen* "to cancel". The highest node contains the length of the parts before and after the LCS. The left node in the second row contains the length of the parts before and after the LCS of *abge* and *ab*. The prefix *sub* indicates that the node is a substitution operation.

principle be applied to any MRI system.

An edit tree $e(\sigma, \tau)$ specifies a transformation from a source string σ to a target string τ (Chrupała, 2008). To compute $e(\sigma, \tau)$, we first determine the longest common substring (LCS) (Gusfield, 1997) between σ and τ and then recursively model the prefix and suffix pairs of the LCS. If the length of LCS is zero for (σ, τ), then $e(\sigma, \tau)$ is simply the substitution operation that replaces σ with τ. Figure 2 shows an example.

Let X be a training set for MRI. For each pair (s, t) of tags, we define:

$$E_{s,t} = \{e' | \exists x \in X : e' = e(x), s = S(x), t = T(x)\}$$

where $S(x)$ and $T(x)$ are source and target tags of x and $e(x)$ is $e(\sigma(x), \tau(x))$, the edit tree that transforms the source form into the target form.

Let ρ be a target form predicted by the MRI system for the source form σ and let s and t be source and target tags. POET does not change ρ if $e(\sigma, \rho) \in E_{s,t}$. Otherwise it replaces ρ with τ:

$$\tau := \begin{cases} \tau' & \text{if } e(\sigma, \tau') \in E_{s,t}, |\rho, \tau'| = 1 \\ \rho & \text{else} \end{cases}$$

where $|\rho, \tau'|$ is the Levenshtein distance. If there are several forms τ' with edit distance 1, we select the one with the most frequent edit tree. Ties are broken randomly.

We observed that MED sometimes makes errors that are close to the target, but differ by one edit operation. Those errors are often not covered by edit trees that are observed in the training data whereas the correct form is. Thus, substituting a form not supported by an observed edit tree with a close one that is supported promises to reduce the error rate.

The effectiveness of POET depends on a training set that is large enough to cover the possible edit trees that can occur in reinflection in a language. Thus, if the training set is not large enough in this respect, then POET will not be beneficial.

3 Experiments

We compare MED with the three models of Dreyer et al. (2008) as well as with two recently proposed models: (i) discriminative string transduction (Durrett and DeNero, 2013; Nicolai et al., 2015), the SIGMORPHON16 baseline, and (ii) Faruqui et al. (2015)'s encoder-decoder model.[3] We call the latter MODEL*TAG as it requires training as many models as there are target tags.

We evaluate MED on two MRI tasks: CELEX and SIGMORPHON16.

CELEX. This task is based on complete inflection tables for German extracted from CELEX. For this experiment we follow Dreyer et al. (2008). We use four pairs of morphological tags and corresponding word forms from the German part of the CELEX morphological database. The 4 different transduction tasks are: 13SIA → 13SKE, 2PIE → 13PKE, 2PKE → z and rP → pA.[4] An example for this task would be to produce the output *gesteuert* (target tag *pA*) for the source *steuert* (source tag *rP*). To do so, the system has to learn that the prefix *ge-*, which is used for many participles in German, has to be added to the beginning of the original word form.

We use the same data splits as Dreyer et al. (2008), dividing the original 2500 samples for each tag into five folds, each consisting of 500 training and 1000 development and 1000 test samples. We train a separate model for each fold and report exact match accuracy, averaged over the five folds, as our final result.

[3] For our experiments we ran the code available at `github.com/mfaruqui/morph-trans`. We used the *enc-dec-attn* model as overall results for the CELEX task were better than with the *sep-morph* model.

[4] 13SIA=1st/3rd sg. ind. past; 13SKE=1st/3rd sg. subjunct. pres.; 2PIE=2nd pl. ind. pres.; 13PKE=1st/3rd pl. subjunct. pres.; 2PKE=2nd. pl. subjunct. pres.; z=infinitive; rP=imperative pl.; pA=past part.

	model	13SIA	2PIE	2PKE	rP
Dreyer	backoff	82.8	88.7	74.7	69.9
	lat-class	84.8	93.6	75.7	81.8
	lat-region	**87.5**	93.4	87.4	**84.9**
	baseline	77.6	**95.1**	82.5	69.6
	MODEL*TAG	76.4	92.1	83.4	81.8
	MED	82.3	94.4	86.8	83.9
	MED+POET	83.9	95.0	**87.6**	84.0

Table 1: Exact match accuracy of MRI on CELEX. Results of (Dreyer et al., 2008)'s model are from their paper; backoff: *ngrams+x* model; lat-class: *ngrams+x+latent class* model; lat-region: *ngrams+x+latent class+latent region* model; baseline: SIGMORPHON16 baseline.

	MED		
	baseline	average	ensemble
Arabic	58.8	83.1 (0.4)	**88.8**
Finnish	64.6	92.5 (0.8)	**95.6**
Georgian	91.5	95.7 (0.3)	**97.3**
German	87.7	92.1 (0.5)	**95.1**
Navajo	60.9	85.0 (1.1)	**91.1**
Russian	85.6	84.2 (0.3)	**88.4**
Spanish	95.6	96.3 (0.3)	**97.5**
Turkish	54.9	94.7 (1.3)	**97.6**

Table 2: Exact match accuracy of MRI on SIGMORPHON16; baseline: SIGMORPHON16 baseline; MED/average: average of five MED models (standard deviation in parentheses); MED/ensemble: majority voting of five MED models.

SIGMORPHON16. This task covers eight languages and does not provide complete paradigms, but only a set of quadruples, each consisting of word form, source tag, target tag and target form. The main difference to CELEX is that the number of tag pairs is large, resulting in much less training data per tag pair. The number of tag pairs varies by language with Georgian being an extreme case; it has 28 tag pairs in dev that appear less than 10 times in train. For each language, we have around 12,800 training and 1600 development samples. We report exact match accuracy on the development set, as the final test data of the shared task is not publically available yet.

4 Results

Table 1 gives CELEX results. MED+POET is better than prior work on one task, close in performance on two and worse by a small amount on the third. Unlike Dreyer et al. (2008)'s models, MED does not use any hand-crafted features. MED's results are weakest on 13SIA. Typical errors on this task include epenthesis (e.g., *zirkle* vs. *zirkele*) and irregular verbs (e.g., *abhing* vs. *abhängte*).

For SIGMORPHON16, Table 2 shows that MED outperforms the baseline for all eight languages. Absolute performance and variance is probably influenced by type of morphology (e.g., templatic vs. agglutinative), regularity of the language, number of different tag pairs and other factors. MED performs well even for complex and diverse languages like Arabic, Finnish, Navajo and Turkish, suggesting that the type of attention-based encoder-decoder we use – single-model, using an explicit morphological representation – is a good choice for MRI.

We do not compare to MODEL*TAG here because it requires training a large number of individual networks. This is a disadvantage compared to MED both in terms of the number of models that need to be trained and in terms of the effective use of the small number of training examples that are available per tag pair.

POET improves the results for all tag pairs for CELEX. However, initial experiments indicated that it is not effective for SIGMORPHON16 because its training sets are not large enough.

5 Analysis

The main innovation of our work is that MED learns a single model of all MRI patterns of a language and thus can transfer what it has learned from one tag pair to another tag pair. Using CELEX, we now analyze how much our design contributes to better performance by conducting two experiments in which we gradually decrease the training set in two different ways. (i) Large general training set. We only reduce the number of training examples available for a tag pair (s, t) and retain all other training examples. (ii) Small training set. We reduce the number of training examples available for all tag pairs, not just for one.

A typical example of the large general training set scenario is that familiar second person forms are rare in genres like encyclopedia and news. So a training set derived from these genres will be large, but it will have very few tag pairs whose target tag is familiar second person.

A typical example of the small training set scenario is that we are dealing with a low-resource language.

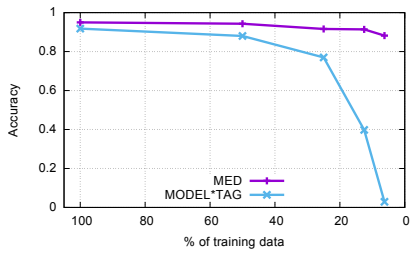


Figure 3: Results for the large general training set experiment: effect of reducing the training set *for only 2PIE → 13PKE* on the accuracy for 2PIE → 13PKE for MED and MODEL*TAG.

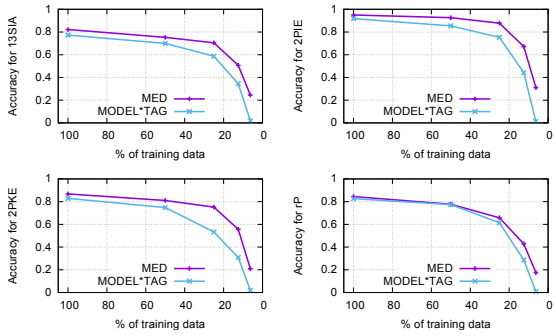


Figure 4: Results for the small training set experiment: effect of reducing the training set *for all tag pairs* on accuracy for MED and MODEL*TAG.

In the following two experiments, we only reduce the training set and do not change the test set.

Large general training set. We iteratively halve the training data for 2PIE → 13PKE until only 6.25% or 32 samples are left. Figure 3 shows that MED performs well even if only 6.25% of the training examples for the tag pair remain. In contrast, MODEL*TAG struggles to generalize correctly. This is due to the fact that we train one single model for all tags, so it can learn from other tags and transfer what it has learned to the tag pair that has a small training set.

Small training set. Figure 4 shows results for reducing the training data equally for all tags. MED performs much better than the baseline for less than 50% of the training data. This can be explained by the fact that MED learns from all given data at once and thus is able to learn common patterns that apply across different tag pairs.

6 Related Work

Earlier work on morphology includes morphological segmentation (Harris, 1955; Hafer and Weiss, 1974; Déjean, 1998) and different approaches for MRI (Ahlberg et al., 2014; Durrett and DeNero, 2013; Eskander et al., 2013; Nicolai et al., 2015). Chrupała (2008) defined edit trees and Chrupała (2008) and Müller et al. (2015) use them for morphological tagging and lemmatization.

In the last years, RNN encoder-decoder models and RNNs in general were applied to several NLP tasks. For example, they proved to be useful for machine translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014), parsing (Vinyals et al., 2015) and speech recognition (Graves and Schmidhuber, 2005; Graves et al., 2013).

MED bears some resemblance to Faruqui et al. (2015)'s work. However, they train one network for every tag pair; this can negatively impact performance for low-resource languages and in general when training data are limited. In contrast, we train a single model for each language. This radically reduces the amount of training data needed for the encoder-decoder because most MRI patterns occur in many tag pairs, so what is learned for one can be transferred to others. To be able to model all tag pairs of the language together, we introduce an explicit morphological representation that enables the attention mechanism of the encoder-decoder to generalize MRI patterns across tag pairs.

7 Conclusion and Future Work

We have presented MED, a language independent neural sequence-to-sequence mapping approach, and POET, a method based on edit trees for correcting the output of an MRI system. MED obtains results comparable to state-of-the-art systems for CELEX and establishes the state-of-the-art for SIGMORPHON16. POET improves results further for large training sets. Our analysis showed that MED outperforms a neural encoder-decoder baseline system by a large margin, especially for small training sets.

In future work, we would like to make POET less dependent on the source tag and thus increase its accuracy for small training sets. Second, we will look into ways of taking advantage of additional information sources including unlabeled corpora.

Acknowledgments

We gratefully acknowledge the financial support of Siemens for this research.

References

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

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Grzegorz Chrupała. 2008. *Towards a machine-learning architecture for lexical functional grammar parsing*. Ph.D. thesis, Dublin City University.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proc. of the 2016 Meeting of SIGMORPHON*.

Hervé Déjean. 1998. Morphemes as necessary concept for structures discovery from untagged corpora. In *Proc. of the Joint Conferences on New Methods in Language Processing and CoNLL*.

Markus Dreyer, Jason R Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proc. of EMNLP*.

Markus Dreyer. 2011. *A non-parametric model for the discovery of inflectional paradigms from plain text using graphical models over strings*. Ph.D. thesis, Johns Hopkins University, Baltimore, MD.

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

Ramy Eskander, Nizar Habash, and Owen Rambow. 2013. Automatic extraction of morphological lexicons from morphologically annotated corpora. In *Proc. of EMNLP*.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2015. Morphological inflection generation using character sequence to sequence learning. *arXiv preprint arXiv:1512.06110*.

Ian Goodfellow, David Warde-farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *Proc. of ICML*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proc of. ICASSP*.

Dan Gusfield. 1997. *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge university press.

Margaret A Hafer and Stephen F Weiss. 1974. Word segmentation by letter successor varieties. *Information storage and retrieval*, 10(11):371–385.

Zellig S Harris. 1955. From phoneme to morpheme. *Language*, 31(2):190–222.

Katharina Kann and Hinrich Schütze. 2016. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *Proc. of the 2016 Meeting of SIGMORPHON*.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Thomas Müller, Ryan Cotterell, and Alexander Fraser. 2015. Joint lemmatization and morphological tagging with lemming. In *Proc. of EMNLP*.

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

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to construct deep recurrent neural networks. In *Proc. of ICLR*.

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

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proc. of NIPS*.

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

Chapter 3

MED: The LMU System for the SIGMORPHON 2016 Shared Task on Morphological Reinflection

MED: The LMU System for the SIGMORPHON 2016 Shared Task on Morphological Reinflection

Katharina Kann and Hinrich Schütze
Center for Information & Language Processing
LMU Munich, Germany
kann@cis.lmu.de

Abstract

This paper presents MED, the main system of the LMU team for the SIGMORPHON 2016 Shared Task on Morphological Reinflection as well as an extended analysis of how different design choices contribute to the final performance. We model the task of morphological reinflection using neural encoder-decoder models together with an encoding of the input as a single sequence of the morphological tags of the source and target form as well as the sequence of letters of the source form. The Shared Task consists of three subtasks, three different tracks and covers 10 different languages to encourage the use of language-independent approaches. MED was the system with the overall best performance, demonstrating our method generalizes well for the low-resource setting of the SIGMORPHON 2016 Shared Task.

1 Introduction

In many areas of natural language processing (NLP) it is important that systems are able to correctly analyze and generate different morphological forms, including previously unseen forms. Two examples are machine translation and question answering, where errors in the understanding of morphological forms can seriously harm performance. Accordingly, learning morphological inflection patterns from labeled data is an important challenge.

The task of morphological inflection (MI) consists of generating an inflected form for a given lemma and target tag. Several approaches have been developed for this, including machine learning models and models that exploit the paradigm structure of language (Ahlberg et al., 2015; Dreyer, 2011; Nicolai et al., 2015). A more complex problem is morphological reinflection (MRI). For this, an inflected form has to be found given another inflected form, a target tag and optionally a source tag.

We use the same approach to both MI and MRI: the character-based and language independent sequence-to-sequence attention model called MED, which stands for *Morphological Encoder-Decoder*. To solve the MRI task, we train one single model on all available source-to-target mappings for each language contained in the training set. This enables the encoder-decoder to learn good parameters for relatively small amounts of training data per target tag already, because most MRI patterns occur in many source-target tag pairs. In our model design, what is learned for one pair can be transferred to others.

The most important point for this is the representation we use for MRI. We encode the input as a single sequence of (i) the morphological tags of the source form, (ii) the morphological tags of the target form and (iii) the sequence of letters of the source form. The output is the sequence of letters of the target form. We train a single generic encoder-decoder per language on this representation that can handle all tag pairs, thus making it possible to make efficient use of the available training data.

The SIGMORPHON 2016 Shared Task on Morphological Reinflection covers both, MI and MRI, for 10 languages as well as different settings and MED outperforms all other systems on all subtasks. The given languages, tracks and tasks will be explained briefly now. For further details on the Shared Task please refer to Cotterell et al. (2016).

Languages. In total, the Shared Task covers 10 languages: Arabic, Finnish, Georgian, German, Hungarian, Maltese, Navajo, Russian, Spanish and Turkish. The training and development datasets

for Hungarian and Maltese were only released at evaluation time.

Tasks. The Shared Task consists of 3 separate tasks with increasing difficulty: task 1 is supposed to be the easiest and task 3 the hardest. The first task consists of mapping a given lemma and target tag to a target form. Task 2 requires the mapping of a given source form, source tag and target tag to a target form. Finally, task 3 consists of finding a target form for a given source form and source tag only.

Tracks. The Shared Task is split into 3 tracks that differ in the information available. The first track is the standard track and requires the solution for each task to use only the training and development data of the current and all lower-numbered tasks, e.g., to use only the data for tasks 1 and 2 for task 2. The restricted track limits the available training and development data to the data belonging to the current task, i.e., data from lower tasks cannot be used, making it impossible to reduce task 2 to task 1 or task 3 to task 2. Track 3 is the bonus track. In this track, all available data per language can be used, including unlabeled corpora which are provided by the task organizers. However, those vary a lot in length, depending on the language. Therefore, we do not make use of them.

In total, there are 90 combinations of languages, tasks and tracks to solve.

The remainder of this paper is organized as follows: In Section 2, our model for the SIGMORPHON 2016 Shared Task is presented. Next, our method to preprocess and thus extend the training data is explained in detail. In Section 4 the final results on the test data of the Shared Task are presented and discussed. Afterwards, we analyze the contribution of different settings and components to the overall performance of our system in detail. Finally, in Section 6, we give information about prior work on topics related to our system.

This paper is mainly concerned with the implementation and analysis of the system we submitted to the Shared Task. In (Kann and Schütze, 2016), we instead focus on the novel aspects of our new method MED and compare its performance to prior work on other MRI benchmarks.

2 System description

Our system for the Shared Task is an encoder-decoder recurrent neural network (RNN), called MED, which stands for *Morphological Encoder-Decoder*. It will be described in detail in this Section.

2.1 Neural network model

Our model is based on the network architecture proposed by Bahdanau et al. (2014) for machine translation.[1] The authors describe the model in detail; unless we explicitly say so in the description of our model below, we use the same network configuration as they do.

Bahdanau et al. (2014)'s model is an extension of the recurrent neural network (RNN) encoder-decoder developed by Cho et al. (2014) and Sutskever et al. (2014). The encoder of the latter consists of a gated RNN (GRU) that reads an input sequence of vectors x and encodes it into a fixed-length context vector c, computing hidden states h_t and c by

$$h_t = f(x_t, h_{t-1}) \tag{1}$$

and

$$c = q(h_1, ..., h_{T_x}) \tag{2}$$

with nonlinear functions f and q. The decoder uses the context vector to predict the output y:

$$p(y) = \prod_{t=1}^{T_y} p(y_t|\{y_1, ..., y_{t-1}\}, c) \tag{3}$$

with $y = (y_1, ..., y_{T_y})$ and each conditional probability being modeled with an RNN as

$$p(y_t|\{y_1, ..., y_{t-1}\}, c) = g(y_{t-1}, s_t, c) \tag{4}$$

where g is a nonlinear function and s_t is the hidden state of the RNN.

Bahdanau et al. (2014) proposed an attention-based version of this model that allows different vectors c_t for each step by automatic learning of an alignment model. They further made the encoder bidirectional. In their model each hidden state h_j at time step j does not only depend on the preceding, but also on the following input:

$$h_j = \left[\overrightarrow{h_j^T}; \overleftarrow{h_j^T}\right]^T \tag{5}$$

[1] Our implementation of MED is based on `https://github.com/mila-udem/blocks-examples/tree/master/machine_translation`.

The formula for $p(y)$ changes accordingly:

$$p(y) = \prod_{t=1}^{T_y} p(y_t|\{y_1, ..., y_{t-1}\}, x) \quad (6)$$

$$= \prod_{t=1}^{T_y} g(y_{t-1}, s_t, c_t) \quad (7)$$

with s_t being an RNN hidden state for time t and c_t being the weighted sum of the annotations $(h_1, ..., h_{T_x})$ produced by the encoder:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \quad (8)$$

The attention weights α_{ij} are calculated for each h_j as

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \quad (9)$$

with

$$e_{ij} = a(s_{i-1}, h_j) \quad (10)$$

a is parametrized as a feedforward neural network and trained jointly with all other components.

More theoretical background is given in (Bahdanau et al., 2014) and a system overview can be seen in Figure 1.

The final model is a multilayer network with a single maxout (Goodfellow et al., 2013) hidden layer that computes the conditional probability of each element in the output sequence (a character in our case, (Pascanu et al., 2014)). As MRI is less complex than machine translation, we reduce the number of hidden units and the embedding size. After initial experiments, we fixed the hyperparameters of our system and did not further adapt them to a specific task or language. Encoder and decoder RNNs have 100 hidden units each. For training, we use stochastic gradient descent, Adadelta (Zeiler, 2012) and a minibatch size of 20. We initialize all weights in the encoder, decoder and the embeddings except for the GRU weights in the decoder with the identity matrix as well as all biases with zero (Le et al., 2015). We train all models for 20 iterations for all combinations of track and task where we cannot extend the training data with our method described in the next section. Otherwise, we train for 10 epochs.[2] We settled on this number in early experimentation because training usually converged before that limit.

MED is an ensemble of five RNN encoder-decoders. The final decision is made by majority voting. In case of a tie, the answer is chosen randomly among the most frequent predictions.

2.2 Input and output format

We define the alphabet Σ_{lang} as the set of characters used in the application language. As each tag combination which describes a source or target form consists of one or more subtags, e.g., "number" or "case", we further define Σ_{src} and Σ_{trg} as the set of morphological subtags seen during training as part of the source tag or the target tag, respectively. Finally, we define S_{start} and S_{end} to be a start and an end symbol. Then each input of our system is of the format $S_{start}\Sigma^+_{src}\Sigma^+_{trg}\Sigma^+_{lang}S_{end}$. In the same way, we define the output format as $S_{start}\Sigma^+_{lang}S_{end}$.

For example, a valid input for German would be *<w> IN=pos=ADJ IN=case=GEN IN=num=PL OUT=pos=ADJ OUT=case=ACC OUT=num=PL i s o l i e r t e r </w>*. The corresponding system output should be *<w> i s o l i e r t e </w>*.[3]

3 Data and training

3.1 Training data enhancement

Since the Shared Task models a low-resource setting, a way to enhance the given training data is highly desirable. We apply three different methods for this, depending on the track and, therefore, depending on the information available. Even though the training data enhancer could be used to increase the amount of available data for other models as well, we expect it to be especially effective with MED. This is due to the fact that MED is able to reuse information from any combination of input and output tag for any other tag pair.

Restricted track. In the restricted track, only training and development data of the respective task and language can be used. This means that there is less information available than in the other two tracks. Therefore, in this track we can only use a very basic enhancement method and we can

[2] For extended data in Maltese we trained only for 6 epochs, due to time constraints.

[3] For task 1 in the restricted and standard track and task 3 throughout all tracks, no source tag is given and we only have one tag combination in the input. Therefore, we do not prepend *IN=* or *OUT=* to the tags. However, internally, this does not make a difference for the model.

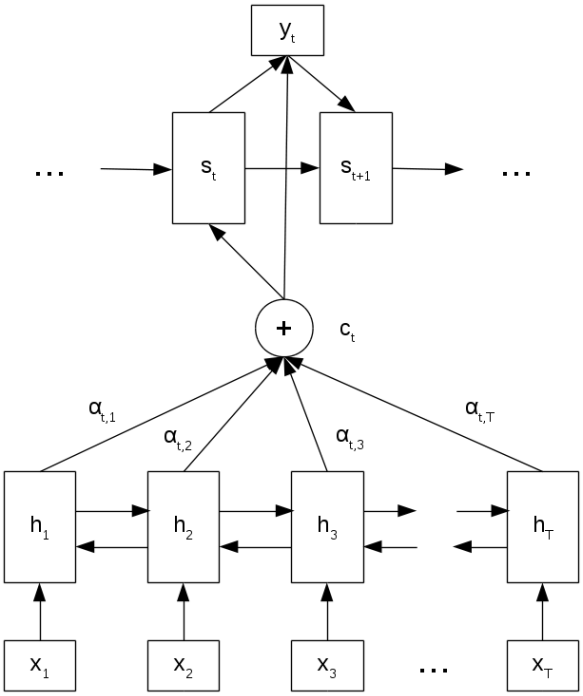


Figure 1: System overview. The input x consists of characters as well as input and output tags. The output y consists of characters only.

only apply it to task 2. The idea the method is based on is that task 2 is symmetric. As described before, the task consists of mapping a triplet of source tag, source form and target tag to a target form. To double the training data it is sufficient to switch the information and thus create a new sample, mapping from target tag, target form and source tag to the source form.

Standard track. The training data enhancement for the standard track combines information from task 1 and task 2 and can therefore, following the Shared Task rules, be used for task 2 and task 3, as only data from lower tasks needs to be accessed. The idea of our enhancement method is that each word form belongs to a certain paradigm which in turn belongs to one single lemma. Therefore, when knowing the lemmas of words, we can group them into paradigms. When having more than one word per paradigm, we can infer the information that all of them can be inflected into each other and thus use them to create new samples. Knowing this, we use task 1 training data to make groups of lemmas and word forms belonging to the same paradigm, keeping the tags. Then, we add all information from task 2 and, knowing that source form and target form always belong to the same lemma, we add both forms with their tags to a group whenever one of them is already in there.[4] Afterwards, we build all combinations of word pairs of each paradigm and, by doing so, create new training data.

This method could be applied even if there was absolutely no overlap between the lemmas in task 1 and task 2. However, it would then be necessary to train a lemmatizer on task 1 data first and lemmatize all words to sort them into paradigms. When doing this, accuracy could be improved by only accepting predictions with a strong confidence and by only accepting new words for a paradigm if the source and the target form of a sample have the same lemma prediction.

Bonus track. In the bonus track, our training data enhancement can also be used for task 1. In order to do so, we first apply the same method as for the standard track to produce the extended training data. However, we additionally change the input format for task 1 such that it resembles the task 2 input, using *LEMMA* as the input tag. In this way, we can apply the task 2 model to task 1 such that task 1 is able to benefit from the additional data as well.

3.2 Description of the final training data

Depending on the complexity of the language and the structure of the datasets we end up with a different amount of final training samples for each language, even though we start with nearly identical training set sizes. We show the final number of samples for task 2 in different tracks in Table 1. As can be seen, the training data enhancement increases the number of samples by a factor between 10 and 80. Out of all languages, the enhancer has the smallest effect for Finnish. Most additional samples are created for Maltese.

3.3 Training

For each of the 10 languages we train one ensemble for each task of the restricted track as well as for each of tasks 2 and 3 of the bonus track. We do not train a separate model for task 1, due to the fact that the same model can be applied to both task 1 and task 2 of the bonus track. In total, we train 50 ensembles, consisting of 250 single models. For our setting, task 1 of the standard track is the same as for the restricted track, while tasks 2 and 3 are the same as for the bonus track.

[4] As for none of the languages task 3 contained new word forms, we did not consider task 3 data here.

Dataset	T2, given no. samples	T2, restricted no. samples	T2, restricted factor	T2, standard no. samples	T2, standard factor
Arabic	14,400	28,800	2	458,814	32
Finnish	14,400	28,800	2	116,206	8
Georgian	14,400	28,800	2	196,396	14
German	14,400	28,800	2	166,148	12
Hungarian	21,600	43,200	2	643,630	30
Maltese	21,600	43,200	2	1,629,446	75
Navajo	14,385	28,770	2	160,332	11
Russian	14,400	28,800	2	129,302	9
Spanish	14,400	28,800	2	211,030	15
Turkish	14,400	28,800	2	392,136	27

Table 1: Number of training samples for task 2 without (given) and with the training data enhancer (restricted and standard track) together with the factor by which the size of the training set increased. Note that the samples for task 2 in the standard track are the same as the samples for task 1 in the bonus track.

Language	Task 1	Task 2	Task 3
Arabic	95.47%	97.38%	96.52%
Finnish	96.80%	97.40%	96.56%
Georgian	98.50%	99.14%	98.87%
German	95.80%	97.45%	95.60%
Hungarian	99.30%	99.67%	99.50%
Maltese	88.99%	88.17%	87.83%
Navajo	91.48%	96.64%	96.20%
Russian	91.46%	91.00%	89.91%
Spanish	98.84%	98.74%	97.96%
Turkish	98.93%	97.94%	99.31%

Table 2: Exact-match accuracy per language for the standard track of the SIGMORPHON 2016 Shared Task.

Language	Task 1	Task 2	Task 3
Arabic	95.47%	91.09%	82.80%
Finnish	96.80%	96.81%	93.18%
Georgian	98.50%	98.50%	96.21%
German	95.80%	96.22%	92.41%
Hungarian	99.30%	99.42%	98.37%
Maltese	88.99%	86.88%	84.25%
Navajo	91.48%	97.81%	83.50%
Russian	91.46%	90.11%	87.13%
Spanish	98.84%	98.45%	96.69%
Turkish	98.93%	98.38%	95.00%

Table 3: Exact-match accuracy per language for the restricted track of the SIGMORPHON 2016 Shared Task.

For each task of the restricted track we train a separate model for 20 epochs. For the bonus track we reduce the number of epochs to 10, because we have much more training data. For Maltese, we reduce it even further to 6 epochs.

Because we do not tune any hyperparameters, we combine the original training and development sets to one big training set. The numbers reported in Table 1 are considering this big dataset.

4 Results on the Shared Task test data

Tables 2, 3 and 4 list the official final results of MED for the SIGMORPHON 2016 Shared Task. Table 2 shows the results of the standard track for which systems are allowed to access the data of the respective task and all lower numbered tasks. Therefore, we can apply our training data extension to tasks 2 and 3, but not to task 1. Because of this, the two higher tasks have the same scores as in the bonus track: we effectively give the same answers. Task 1, in turn, is the same for the standard and the restricted track, leading to the same numbers in Tables 2 and 3.

Language	Task 1	Task 2	Task 3
Arabic	98.25%	97.38%	96.25%
Finnish	97.30%	97.40%	96.56%
Georgian	99.20%	99.14%	98.87%
German	97.38%	97.45%	95.60%
Hungarian	99.69%	99.67%	99.50%
Maltese	88.53%	88.17%	87.83%
Navajo	98.03%	96.64%	96.20%
Russian	92.15%	91.00%	89.91%
Spanish	99.05%	98.74%	97.96%
Turkish	97.49%	97.94%	99.31%

Table 4: Exact-match accuracy per language for the bonus track of the SIGMORPHON 2016 Shared Task.

For ease of exposition, we will mostly compare the restricted and the bonus track as the standard track can be considered a mixture of those two. For most tasks and languages the accuracy is higher in the bonus than in the restricted track. This is easy to explain as MED has more data to train on (task 1 information for tasks 2 and 3 and task 2 information for task 1). The exception is Navajo: For task 2 the accuracy is higher in the bonus track than in the restricted track. We leave an investigation of this for future work.

Our training data enhancer – which is the only difference between the bonus and the restricted track as we do not use the provided unlabeled corpora – is clearly effective: For Arabic, for example, it leads to 13.72% improvement in performance for task 3. For Turkish, the accuracy for task 3 increases by 4.31%. Those are also the lan-

guages for which the training data enhancement was very effective as can be seen in Table 1. That Maltese does not improve so much even though we use a lot more training data is most likely due to the shorter training: we trained only for 6 epochs instead of 10, because of time constraints.

As expected, the scores for task 3 are worse than or at most comparable to the scores for task 2 in all tracks. This is due to the fact that task 3 does not provide a source tag, so less information is available. However, it seems that this information was not much needed as the improvement when adding it is minor. The better result for task 3 for Turkish compared to task 2 in the bonus track may be due to randomness during training – like the order of samples in the training data – as it is below 1.5%.

It may be surprising at first that the results for task 1 are not always better than the results for task 2. This is the case, for example, in the restricted track for Finnish, Georgian, Hungarian and Navajo. As the organizers describe on the Shared Task's homepage, they expect task 1 to be the easiest. Our guess would be that the model has more information in total for task 2 as more forms are given per paradigm. Additionally, task 2 is symmetric; this makes it possible to use twice the training data, as described in Section 3.

5 System Analysis

To analyze which design choices are important and how they influence the performance of MED we conduct several experiments, always keeping all but the investigated design choice fixed to the settings described in Section 2. To make the experiments clearer, we limit them to one combination of task, track and language: Unless mentioned otherwise, we perform all experiments described in this section on task 2 of the restricted track for Russian. For the experiments in this section, the system is trained on training data only and evaluated on the development set. The training data enhancement is not used in this analysis.

5.1 Analysis 1: Number of hidden units in encoder and decoder

In its original configuration MED has 100 hidden units in both the encoder and the decoder. This number was found to be good during initial experiments. However, we want to investigate how the number of hidden units in the RNNs can effect the final accuracy on an MRI task. Therefore, we train one ensemble for each of 50, 100, 200 and 400 hidden units. To reduce the number of possible different options and because it agrees with MED's original configuration, we define the numbers of hidden units in encoder and decoder to be equal.

Number of hidden units	Exact-match accuracy
50	86.2%
100	88.4%
200	87.2%
400	87.3%

Table 5: Performance of MED for different numbers of hidden units in the encoder and decoder.

Embedding size	Exact-match accuracy
100	86.7%
200	87.3%
300	88.4%
400	90.0%
500	90.3%

Table 6: Performance of MED for different embedding dimensions in the encoder and decoder.

The evaluation in Table 5 shows that the best accuracy is obtained for 100 hidden units. Lower results for fewer hidden units indicate that the model does not have enough capacity to learn the patterns in the data well. Lower results for more hidden units indicate that the model is overfitting the training data.

5.2 Analysis 2: Size of the embeddings

We chose 300 to be the size of the character and tag embeddings in our model for the Shared Task. In this analysis, we want to systematically investigate how MED performs for different embedding sizes for the encoder and decoder embeddings. We train the model with embeddings of the sizes 100, 200, 300, 400 and 500 and report the resulting accuracies in Table 6.

The results show that the bigger the embeddings get the more the perfomance improves. The best accuracy is reached for 500-dimensional embeddings, i.e., the biggest embeddings in this analysis. This suggests that we might have improved our final results in the Shared Task even further by using embeddings of a higher dimensionality. However, this is also a trade-off between a gain in accuracy and longer training time. Keeping in mind that we had to train many single models, 300 was a reasonable choice for the embedding size, with only 1.9% loss of perfomance compared to 500-dimensional embeddings.

Initialization	Exact-match accuracy
Identity	90.5%
Identity + orthogonal	88.4%
Gaussian + orthogonal	89.7%

Table 7: Performance of MED for different initialization types.

5.3 Analysis 3: Initialization

For the Shared Task, most weights of MED are initialized with the identitiy matrix. An exception to this are the weights in the decoder GRU which are initialized using a random orthogonal matrix. All biases are initialized to zero. We now compare how MED's final performance depends on the type of initialization. For this, we train two additional models: (i) we initialize all weights with the identitiy matrix and (ii) we initialize all weights except for the weights in the decoder GRU from a Gaussian distribution. The weights in the decoder GRU are again initialized with a random orthogonal matrix.

The final accuracy of the three models can be seen in Table 7. The random intialization leads to better results than intitializing with the identity matrix together with a random orthogonal matrix. However, the highest accuracy is reached by initializing *all* weights with identity matrices. In fact, the results are 2.1% better than MED's original performance. Thus, we would recommend this initialization for future use of our model.

5.4 Analysis 4: One embedding per tag vs. one embedding per tag combination

To keep the model flexible to handle tag combinations not present in the training set, we split each tag combination into single tags, e.g., *pos=ADJ,case=ACC,gen=FEM,num=SG* becomes *pos=ADJ*, *case=ACC*, *gen=FEM* and *num=SG* with each part having its own embedding which is fed into the model.

We now compare to the performance of a representation in which tags are "fused" and each tag combination has only one single embedding. As this is one of the most important design choices for MED, we do this analysis for several languages and additionally report the number of tag combinations that are not seen during training.

Table 8 shows that unknown tag combinations are generally not a problem with the exception of Maltese. Nevertheless, there is a considerable decrease in performance. The difference is especially big for languages with a lower performance like Arabic, Maltese, Navajo and Russian. Languages with a general high accuracy do not lose much accuracy when using one embedding per tag combination. We hypothesize that the patterns of these languages are easy enough to even be learned with a harder representation. Overall, it seems that our representation with split-up tag combinations is the better choice for MRI and might even be a key component for MED's success in the Shared Task.

Language	MED	MED-tag-comb.	Unk.
Arabic	88.8%	83.4%	0
Finnish	95.6%	95.2%	1
Georgian	97.3%	95.6%	0
German	95.1%	93.5%	1
Hungarian	99.3%	99.3%	0
Maltese	85.7%	77.1%	151
Navajo	91.1%	83.4%	1
Russian	88.4%	86.8%	1
Spanish	97.5%	97.0%	0
Turkish	97.6%	95.9%	2

Table 8: Exact match accuracy for the standard representation (MED) as well as the representation with one embedding per tag combination (MED-tag-comb) per language. The last column shows the number of samples that contain tag combinations that appear in dev but not in train, either for the source or the target form.

Tag order type	Exact-match accuracy
MED	88.4%
MED-perm	86.4%

Table 9: Performance of MED when training on samples with tags in always the same order (MED) and samples where the tags are permuted inside each combination (MED-perm).

5.5 Analysis 5: The order of tags

In the representation we feed to MED, the order of single tags inside a tag combination is always fixed. We now investigate how much influence this has on the final performance of the model; i.e., we ask: is MRI harder or easier to learn if we permutate the morphological tags? For this analysis, we randomly shuffle the tags of each combination in the training and development data (while still using the development set for testing).

Table 9 shows that learning seems to be easier for non-permuted tags. Indeed, when keeping the order of tags fixed, the system performance is 2% better than for the random tag order. However, the difference is not big. This might actually be different for languages other than Russian as we did not investigate from a linguistic point of view if the order matters contentwise for any of the languages.

6 Related Work

Prior work on morphology includes morphological segmentation (Harris, 1955; Hafer and Weiss, 1974; Déjean, 1998), different approaches for MRI (Ahlberg et al., 2014; Durrett and DeNero, 2013; Eskander et al., 2013; Nicolai et al., 2015). and work on morphological tagging and lemmatization (Müller et al., 2015).

RNN encoder-decoder models, gated RNNs in general as well as LSTMs were applied to several NLP tasks including some on morphology like morphological segmentation (Wang et al., 2016) during the last years. Other tasks they proved to be useful for are machine translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014), parsing (Vinyals et al., 2015) or speech recognition (Graves and Schmidhuber, 2005; Graves et al., 2013).

The most similar work to ours was probably the one by Faruqui et al. (2015). Indeed, MED's design is very close to their model. However, they trained one network for every tag pair; this can negatively impact performance in a setting with limited training data like the SIGMORPHON 2016 Shared Task. In contrast, we train a single model for each language. This radically reduces the amount of training data needed for the encoder-decoder because most MRI patterns occur in many tag pairs, so what is learned for one can be transferred to others. In order to model all tag pairs of the language together, we introduce an explicit morphological representation that enables the attention mechanism of the encoder-decoder to generalize MRI patterns across tag pairs.

7 Conclusion

In this paper we described MED, our system for the SIGMORPHON 2016 Shared Task on Morphological Reinflection as well as a training data enhancement method based on paradigms. MED is a powerful character-based encoder-decoder RNN and its architecture is completely language-independent, such that we trained the models for all 10 languages of the Shared Task using the same hyperparameters. MED establishes the state of the art for the SIGMORPHON 2016 Shared Task, scoring first in all of the 90 subtasks of the final evaluation.

Furthermore, we presented an extended analysis, evaluating different design choices for MED. The results show that most of our initial settings were good choices, especially the representation of morphological tags. However, it might be possible to further improve MED's performance increasing the size of the used embeddings and choosing another initialization.

Acknowledgments

We are grateful to MILA (`https://mila.umontreal.ca`) for making their neural machine translation model available to us. We further acknowledge the financial support of Siemens for this research.

References

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

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proc. of the 2016 Meeting of SIGMORPHON*.

Hervé Déjean. 1998. Morphemes as necessary concept for structures discovery from untagged corpora. In *Proc. of the Joint Conferences on New Methods in Language Processing and CoNLL*.

Markus Dreyer. 2011. *A non-parametric model for the discovery of inflectional paradigms from plain text using graphical models over strings*. Ph.D. thesis, Johns Hopkins University, Baltimore, MD.

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

Ramy Eskander, Nizar Habash, and Owen Rambow. 2013. Automatic extraction of morphological lexicons from morphologically annotated corpora. In *Proc. of EMNLP*.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2015. Morphological inflection generation using character sequence to sequence learning. *arXiv preprint arXiv:1512.06110*.

Ian Goodfellow, David Warde-farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *Proc. of ICML*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proc. of ICASSP*.

Margaret A Hafer and Stephen F Weiss. 1974. Word segmentation by letter successor varieties. *Information storage and retrieval*, 10(11):371–385.

Zellig S Harris. 1955. From phoneme to morpheme. *Language*, 31(2):190–222.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proc. of ACL*.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proc. of EMNLP*.

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

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to construct deep recurrent neural networks. In *Proc. of ICLR*.

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

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proc. of NIPS*.

Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. 2016. Morphological segmentation with window LSTM neural networks. In *Proc. of AAAI*.

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

Chapter 4

One-Shot Neural Cross-Lingual Transfer for Paradigm Completion

One-Shot Neural Cross-Lingual Transfer for Paradigm Completion

Katharina Kann
CIS
LMU Munich, Germany
kann@cis.lmu.de

Ryan Cotterell
Department of Computer Science
Johns Hopkins University, USA
ryan.cotterell@jhu.edu

Hinrich Schütze
CIS
LMU Munich, Germany
inquiries@cislmu.org

Abstract

We present a novel cross-lingual transfer method for paradigm completion, the task of mapping a lemma to its inflected forms, using a neural encoder-decoder model, the state of the art for the monolingual task. We use labeled data from a high-resource language to increase performance on a low-resource language. In experiments on 21 language pairs from four different language families, we obtain up to 58% higher accuracy than without transfer and show that even zero-shot and one-shot learning are possible. We further find that the degree of language relatedness strongly influences the ability to transfer morphological knowledge.

1 Introduction

Low-resource natural language processing (NLP) remains an open problem for many tasks of interest. Furthermore, for most languages in the world, high-cost linguistic annotation and resource creation are unlikely to be undertaken in the near future. In the case of morphology, out of the 7000 currently spoken (Lewis, 2009) languages, only about 200 have computer-readable annotations (Sylak-Glassman et al., 2015) – although morphology is easy to annotate compared to syntax and semantics. *Transfer learning* is one solution to this problem: it exploits annotations in a high-resource language to train a system for a low-resource language. In this work, we present a method for cross-lingual transfer of inflectional morphology using an encoder-decoder recurrent neural network (RNN). This allows for the development of tools for computational morphology with limited annotated data.

In many languages, individual lexical entries may be realized as distinct inflections of a single lemma depending on the syntactic context. For example, the 3SgPresInd of the English verbal lemma *to bring* is *brings*. In morphologically rich languages, a lemma can have hundreds of individual forms. Thus, both generation and analysis of such morphological inflections are active areas of research in NLP and morphological processing has been shown to be a boon to several other down-stream applications, e.g., machine translation (Dyer et al., 2008), speech recognition (Creutz et al., 2007), parsing (Seeker and Çetinoğlu, 2015), keyword spotting (Narasimhan et al., 2014) and word embeddings (Cotterell et al., 2016b), *inter alia*. In this work, we focus on paradigm completion, a form of morphological generation that maps a given lemma to a target inflection, e.g., (*bring*, Past) $\mapsto$ *brought* (with Past being the target tag).

	Present Indicative		Past Indicative	
	Sg	Pl	Sg	Pl
1	*sueño*	*soñamos*	*soñé*	*soñamos*
2	*sueñas*	*soñáis*	*soñaste*	*soñasteis*
3	*sueña*	*sueñan*	*soñó*	*soñaron*

Table 1: Partial inflection table for the Spanish verb *soñar*.

RNN sequence-to-sequence models (Sutskever et al., 2014; Bahdanau et al., 2015) are the state of the art for paradigm completion (Faruqui et al., 2016; Kann and Schütze, 2016a; Cotterell et al., 2016a). However, these models require a large amount of data to achieve competitive performance; this makes them unsuitable for out-of-the-box application to paradigm completion in the low-resource scenario. To mitigate this, we consider transfer learning: we train an end-to-end neural system jointly with limited data from a low-resource language and a larger amount of data from a high-resource language. This technique allows

the model to apply knowledge distilled from the high-resource training data to the low-resource language as needed.

We conduct experiments on 21 language pairs from four language families, emulating a low-resource setting. Our results demonstrate successful transfer of morphological knowledge. We show improvements in accuracy and edit distance of up to 58% (accuracy) and 4.62 (edit distance) over the same model with only in-domain language data on the paradigm completion task. We further obtain up to 44% (resp. 14%) improvement in accuracy for the one-shot (resp. zero-shot) setting, i.e., one (resp. zero) in-domain language sample per target tag. We also show that the effectiveness of morphological transfer depends on language relatedness, measured by lexical similarity.

2 Inflectional Morphology and Paradigm Completion

Many languages exhibit inflectional morphology, i.e., the form of an individual lexical entry mutates to show properties such as person, number or case. The citation form of a lexical entry is referred to as the **lemma** and the collection of its possible inflections as its **paradigm**. Tab. 1 shows an example of a partial paradigm; we display several forms for the Spanish verbal lemma *soñar*. We may index the entries of a paradigm by a **morphological tag**, e.g., the 2SgPresInd form *sueñas* in Tab. 1. In generation, the speaker must select an entry of the paradigm given the form's context. In general, the presence of rich inflectional morphology is problematic for NLP systems as it greatly increases the token-type ratio and, thus, word form sparsity.

An important task in inflectional morphology is **paradigm completion** (Durrett and DeNero, 2013; Ahlberg et al., 2014; Nicolai et al., 2015; Cotterell et al., 2015; Faruqui et al., 2016). Its goal is to map a lemma to all individual inflections, e.g., (*soñar*, 1SgPresInd) $\mapsto$ *sueño*. There are good solutions for paradigm completion when a large amount of annotated training data is available (Cotterell et al., 2016a).[1] In this work, we address the low-resource setting, a yet unsolved challenge.

[1]The SIGMORPHON 2016 shared task (Cotterell et al., 2016a) on morphological reinflection, a harder generalization of paradigm completion, found that $\geq 98\%$ accuracy can be achieved in many languages with neural sequence-to-sequence models, improving the state of the art by 10%.

2.1 Transferring Inflectional Morphology

In comparison to other NLP annotations, e.g., part-of-speech (POS) and named entities, morphological inflection is especially challenging for transfer learning: we can define a universal set of POS tags (Petrov et al., 2012) or of entity types (e.g., coarse-grained types like *person* and *location* or fine-grained types (Yaghoobzadeh and Schütze, 2015)), but inflection is much more language-specific. It is infeasible to transfer morphological knowledge from Chinese to Portuguese as Chinese does not use inflected word forms. Transferring named entity recognition, however, among Chinese and European languages works well (Wang and Manning, 2014a). But even transferring inflectional paradigms from morphologically rich Arabic to Portuguese seems difficult as the inflections often mark dissimilar subcategories. In contrast, transferring morphological knowledge from Spanish to Portuguese, two languages with similar conjugations and 89% lexical similarity, appears promising. Thus, we conjecture that transfer of inflectional morphology is only viable among *related languages*.

2.2 Formalization of the Task

We now offer a formal treatment of the cross-lingual paradigm completion task and develop our notation. Let Σ_ℓ be a discrete alphabet for language ℓ and let $\mathcal{T}_\ell$ be a set of morphological tags for ℓ. Given a lemma w_ℓ in ℓ, the morphological paradigm (inflectional table) π can be formalized as a set of pairs

$$\pi(w_\ell) = \Big\{ \big(f_k[w_\ell], t_k\big) \Big\}_{k \in T(w_\ell)} \quad (1)$$

where $f_k[w_\ell] \in \Sigma_\ell^+$ is an inflected form, $t_k \in \mathcal{T}_\ell$ is its morphological tag and $T(w_\ell)$ is the set of slots in the paradigm; e.g., a Spanish paradigm is:

$$\pi(\text{soñar}) = \Big\{ \big(\textit{sueño}, \textsc{1SgPresInd}\big), \ldots, \big(\textit{soñaran}, \textsc{3PlPastSbj}\big) \Big\}$$

Paradigm completion consists of predicting missing slots in the paradigm $\pi(w_\ell)$ of a given lemma w_ℓ.

In cross-lingual paradigm completion, we consider a *high-resource source language* ℓ_s (lots of training data available) and a *low-resource target language* ℓ_t (little training data available). We denote the source training examples as $\mathcal{D}_s$ (with $|\mathcal{D}_s| = n_s$) and the target training examples as

$\mathcal{D}_t$ (with $|\mathcal{D}_t| = n_t$). The goal of cross-lingual paradigm completion is to populate paradigms in the low-resource target language with the help of data from the high-resource source language, using only few in-domain examples.

3 Cross-Lingual Transfer as Multi-Task Learning

We describe our probability model for morphological transfer using terminology from multi-task learning (Caruana, 1997; Collobert et al., 2011). We consider two tasks, training a paradigm completor (i) for a high-resource language and (ii) for a low-resource language. We want to train jointly, so we reap the benefits of having related languages. Thus, we define the log-likelihood as

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(k,w_{\ell_t})\in\mathcal{D}_t} \log p_{\boldsymbol{\theta}}\left(f_k[w_{\ell_t}] \mid w_{\ell_t}, t_k, \lambda_{\ell_t}\right) \quad (2)$$
$$+ \sum_{(k,w_{\ell_s})\in\mathcal{D}_s} \log p_{\boldsymbol{\theta}}(f_k[w_{\ell_s}] \mid w_{\ell_s}, t_k, \lambda_{\ell_s})$$

where we tie parameters $\boldsymbol{\theta}$ for the two languages together to allow the transfer of morphological knowledge between languages. The λs are special language tags, cf. Sec. 3.2. Each probability distribution $p_{\boldsymbol{\theta}}$ defines a distribution over all possible realizations of an inflected form, i.e., a distribution over Σ^*. For example, consider the related Romance languages Spanish and French; focusing on one term from each of the summands in Eq. (2) (the past participle of the translation of *to visit* in each language), we arrive at

$$\mathcal{L}_{\text{visit}}(\boldsymbol{\theta}) = \log p_{\boldsymbol{\theta}}(\textit{visitado} \mid \textit{visitar}, \textsf{PastPart}, \textsf{ES})$$
$$+ \log p_{\boldsymbol{\theta}}(\textit{visité} \mid \textit{visiter}, \textsf{PastPart}, \textsf{FR}) \quad (3)$$

Our *cross-lingual* setting forces both transductions to share part of the parameter vector $\boldsymbol{\theta}$, to represent morphological regularities between the two languages in a common embedding space and, thus, to enable morphological transfer. This is no different from *monolingual* multi-task settings, e.g., jointly training a parser and tagger for transfer of syntax.

Based on recent advances in neural transducers, we parameterize each distribution as an encoder-decoder RNN, as in (Kann and Schütze, 2016b). In their setup, the RNN encodes the input and predicts the forms in a *single* language. In contrast, we force the network to predict *two or more* languages.

3.1 Encoder-Decoder RNN

We parameterize the distribution $p_{\boldsymbol{\theta}}$ as an encoder-decoder gated RNN (GRU) with attention (Bahdanau et al., 2015), the state-of-the-art solution for the monolingual case (Kann and Schütze, 2016b). A bidirectional gated RNN encodes the input sequence (Cho et al., 2014) – the concatenation of (i) the language tag, (ii) the morphological tag of the form to be generated and (iii) the characters of the input word – represented by embeddings. The input to the decoder consists of concatenations of $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$, the forward and backward hidden states of the encoder. The decoder, a unidirectional RNN, uses attention: it computes a weight α_i for each h_i. Each weight reflects the importance given to that input position. Using the attention weights, the probability of the output sequence given the input sequence is:

$$p(y \mid x_1, \ldots, x_{|X|}) = \prod_{t=1}^{|Y|} g(y_{t-1}, s_t, c_t) \quad (4)$$

where $y = (y_1, \ldots, y_{|Y|})$ is the output sequence (a sequence of $|Y|$ characters), $x = (x_1, \ldots x_{|X|})$ is the input sequence (a sequence of $|X|$ characters), g is a non-linear function, s_t is the hidden state of the decoder and c_t is the sum of the encoder states h_i, weighted by attention weights $\alpha_i(s_{t-1})$ which depend on the decoder state:

$$c_t = \sum_{i=1}^{|X|} \alpha_i(s_{t-1}) h_i \quad (5)$$

Fig. 1 shows the encoder-decoder. See Bahdanau et al. (2015) for further details.

3.2 Input Format

Each source form is represented as a sequence of characters; each character is represented as an embedding. In the same way, each source tag is represented as a sequence of subtags, and each subtag is represented as an embedding. More formally, we define the alphabet $\Sigma = \cup_{\ell\in L}\Sigma_\ell$ as the set of characters in the languages in L, with L being the set of languages in the given experiment. Next, we define $\mathcal{S}$ as the set of subtags that occur as part of the set of morphological tags $\mathcal{T} = \cup_{\ell\in L}\mathcal{T}_\ell$; e.g., if 1SgPresInd $\in \mathcal{T}$, then 1, Sg, Pres, Ind $\in \mathcal{S}$. Note that the set of subtags $\mathcal{S}$ is defined as attributes from the UNIMORPH schema (Sylak-Glassman, 2016) and, thus, is universal across languages; the schema is

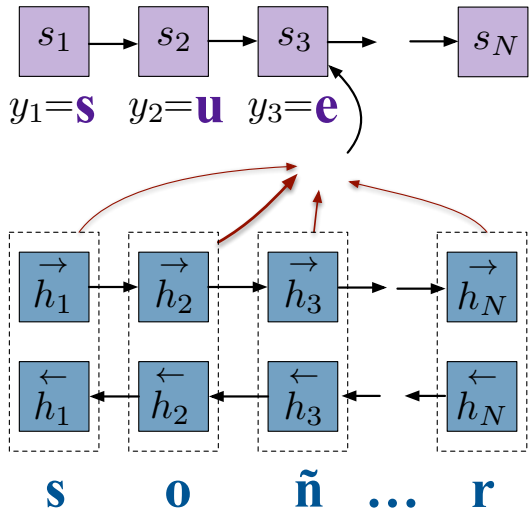


Figure 1: Encoder-decoder RNN for paradigm completion. The lemma *soñar* is mapped to a target form (e.g., *sueña*). For brevity, language and target tags are omitted from the input. Thickness of red arrows symbolizes the degree to which the model attends to the corresponding hidden state of the encoder.

derived from research in linguistic typology.[2] The format of the input to our system is $\mathcal{S}^+\Sigma^+$. The output format is Σ^+. Both input and output are padded with distinguished BOW and EOW symbols.

What we have described is the representation of Kann and Schütze (2016b). In addition, we preprend a symbol $\lambda \in L$ to the input string (e.g., λ = Es, also represented by an embedding), so the RNN can handle multiple languages simultaneously and generalize over them. Thus, our final input is of the form $\lambda\mathcal{S}^+\Sigma^+$.

4 Languages and Language Families

To verify the applicability of our method to a wide range of languages, we perform experiments on example languages from several different families.

Romance languages, a subfamily of Indo-European, are widely spoken, e.g., in Europe and Latin America. Derived from the common ancestor Vulgar Latin (Harris and Vincent, 2003), they share large parts of their lexicon and inflectional morphology; we expect knowledge among them to be easily transferable.

[2]Note that while the subtag set is universal, *which* subtags a language actually uses is language-specific; e.g., Spanish does not mark animacy as Russian does. We contrast this with the universal POS set (Petrov et al., 2012), where it is more likely that we see all 17 tags in most languages.

	PT	CA	IT	FR
similarity to ES	89%	85%	82%	75%

Table 2: Lexical similarities for Romance (Lewis, 2009).

We experiment on Catalan, French, Italian, Portuguese and Spanish. Tab. 2 shows that Spanish – which takes the role of the low-resource language in our experiments – is closely related with the other four, with Portuguese being most similar. We hypothesize that the transferability of morphological knowledge between source and target corresponds to the degree of lexical similarity; thus, we expect Portuguese and Catalan to be more beneficial for Spanish than Italian and French.

The Indo-European **Slavic language family** has its origin in eastern-central Europe (Corbett and Comrie, 2003). We experiment on Bulgarian, Macedonian, Russian and Ukrainian (Cyrillic script) and on Czech, Polish and Slovene (Latin script). Macedonian and Ukranian are low-resource languages, so we assign them the low-resource role. For Romance and for Uralic, we experiment with groups containing three or four source languages. To arrive at a comparable experimental setup for Slavic, we run two experiments, each with three source and one target language: (i) from Russian, Bulgarian and Czech to Macedonian; and (ii) from Russian, Polish and Slovene to Ukrainian.

We hope that the paradigm completor learns similar embeddings for, say, the characters "e" in Polish and "є" in Ukrainian. Thus, the use of two scripts in Slavic allows us to explore transfer across different alphabets.

We further consider a non-Indo-European language family, the **Uralic languages**. We experiment on the three most commonly spoken languages – Finnish, Estonian and Hungarian (Abondolo, 2015) – as well as Northern Sami, a language used in Northern Scandinavia. While Finnish and Estonian are closely related (both are members of the Finnic subfamily), Hungarian is a more distant cousin. Estonian and Northern Sami are low-resource languages, so we assign them the low-resource role, resulting in two groups of experiments: (i) Finnish, Hungarian and Estonian to Northern Sami; (ii) Finnish, Hungarian and Northern Sami to Estonian.

Arabic (baseline) is a Semitic language (part of the Afro-Asiatic family (Hetzron, 2013)) that is

spoken in North Africa, the Arabian Peninsula and other parts of the Middle East. It is unrelated to all other languages used in this work. Both in terms of form (new words are mainly built using a templatic system) and categories (it has tags such as construct state), Arabic is very different. Thus, we do not expect it to support morphological knowledge transfer and use it as a baseline for all target languages.

5 Experiments

We run four experiments on 21 distinct pairings of languages to show the feasibility of morphological transfer and analyze our method. We first discuss details common to all experiments.

We keep **hyperparameters** during all experiments (and for all languages) fixed to the following values. Encoder and decoder RNNs each have 100 hidden units and the size of all subtag, character and language embeddings is 300. For training we use ADADELTA (Zeiler, 2012) with minibatch size 20. All models are trained for 300 epochs. Following Le et al. (2015), we initialize all weights in the encoder, decoder and the embeddings except for the GRU weights in the decoder to the identity matrix. Biases are initialized to zero.

Evaluation metrics: (i) 1-best accuracy: the percentage of predictions that match the true answer exactly; (ii) average edit distance between prediction and true answer. The two metrics differ in that accuracy gives no partial credit and incorrect answers may be drastically different from the annotated form without incurring additional penalty. In contrast, edit distance gives partial credit for forms that are closer to the true answer.

5.1 Exp. 1: Transfer Learning for Paradigm Completion

In this experiment, we investigate to what extent our model transfers morphological knowledge from a high-resource source language to a low-resource target language. We experimentally answer three questions. (i) Is transfer learning possible for morphology? (ii) How much annotated data do we need in the low-resource target language? (iii) How closely related must the two languages be to achieve good results?

Data. Based on complete inflection tables from unimorph.org (Kirov et al., 2016), we create datasets as follows. Each training set consists of 12,000 samples in the high-resource source language and $n_t \in \{50, 200\}$ samples in the low-resource target language. We create target language dev and test sets of sizes 1600 and 10,000, respectively.[3] For Romance and Arabic, we create learning curves for $n_t \in \{100, 400, 800, 1600, 3200, 6400, 12000\}$. Due to the data available to us, we use *only* verbs for the Romance and Uralic language families, but nouns, verbs and adjectives for the Slavic language family and Arabic. Lemmata and inflections are randomly selected from all available paradigms.

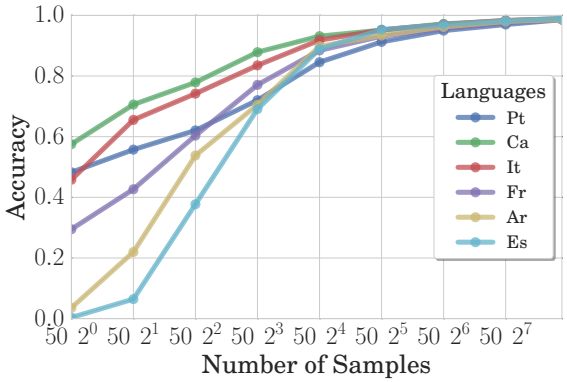


Figure 2: Learning curves showing the accuracy on Spanish test when training on language $\lambda \in \{$PT, CA, IT, FR, AR, ES$\}$. Except for λ=ES, each model is trained on 12,000 samples from λ and "Number of Samples" (x-axis) of Spanish.

Results and Discussion. Tab. 3 shows the effectiveness of transfer learning. There are *two* baselines. (i) "0": no transfer, i.e., we consider only in-domain data; (ii) "AR": Arabic, which is unrelated to all target languages.

With the exception of the 200 sample case of ET→SME, cross-lingual transfer is always better than the two baselines; the maximum improvement is 0.58 (0.58 vs. 0.00) in accuracy for the 50 sample case of CA→ES. More closely related source languages improve performance more than distant ones. French, the Romance language least similar to Spanish, performs worst for →ES. For the target language Macedonian, Bulgarian provides most benefit. This can again be explained by similarity: Bulgarian is closer to Macedonian than the other languages in this group. The best result for Ukrainian is RU→UK. Unlike Polish and Slowenian, Russian is the only language in this group that uses the same script as Ukrainian, showing

[3] For Estonian, we use 7094 (not 12,000) train and 5000 (not 10,000) test samples as more data is unavailable.

		Romance						Slavic I					Slavic II					Uralic I					Uralic II				
source		0	AR	PT	CA	IT	FR	0	AR	RU	BG	CS	0	AR	RU	PL	SL	0	AR	FI	HU	ET	0	AR	FI	HU	SME
target		→ES						→MK					→UK					→SME					→ET				
50	acc ↑	0.00	0.04	0.48	**0.58**	0.46	0.29	0.00	0.00	0.23	**0.47**	0.13	0.01	0.01	**0.47**	0.16	0.07	0.00	0.01	**0.07**	0.05	0.03	0.02	0.01	**0.35**	0.21	0.17
	ED ↓	5.42	4.06	0.85	**0.80**	1.15	1.82	5.71	5.59	1.61	**0.87**	2.32	5.23	4.80	**0.77**	2.14	3.12	6.21	5.47	**2.88**	3.46	3.71	4.50	4.51	**1.55**	2.19	2.60
200	acc ↑	0.38	0.54	0.62	**0.78**	0.74	0.60	0.21	0.40	0.62	**0.77**	0.57	0.16	0.21	**0.64**	0.55	0.50	0.13	0.24	0.26	**0.28**	0.13	0.34	0.53	**0.74**	0.71	0.66
	ED ↓	1.37	0.87	0.57	**0.39**	0.44	0.82	1.93	1.12	0.68	**0.36**	0.72	2.09	1.60	**0.49**	0.73	0.82	2.94	1.89	1.78	**1.61**	2.46	1.47	0.98	**0.41**	0.48	0.62

Table 3: Accuracy (acc; the higher the better; indicated by ↑) and edit distance (ED; the lower the better; indicated by ↓) of cross-lingual transfer learning for paradigm completion. The target language is indicated by "→", e.g., it is Spanish for "→ES". Sources are indicated in the row "source"; "0" is the monolingual case. Except for Estonian, we train on $n_s = 12{,}000$ source samples and $n_t \in \{50, 200\}$ target samples (as indicated by the row). There are *two* baselines in the table. (i) "0": no transfer, i.e., we consider only in-domain data; (ii) "AR": the Semitic language Arabic is unrelated to all target languages and functions as a dummy language that is unlikely to provide relevant information. All languages are denoted using the official codes (SME=Northern Sami).

the importance of the alphabet for transfer. Still, the results also demonstrate that transfer works across alphabets (although not as well); this suggests that similar embeddings for similar characters have been learned. Finnish is the language that is closest to Estonian and it again performs best as a source language for Estonian. For Northern Sami, transfer works least well, probably because the distance between sources and target is largest in this case. The distance of the Sami languages from the Finnic (Estonian, Finnish) and Ugric (Hungarian) languages is much larger than the distances within Romance and within Slavic. However, even for Northern Sami, the worst performing language, adding an additional language is still always beneficial compared to the monolingual baseline.

Learning curves for Romance and Arabic further support our finding that language similarity is important. In Fig. 2, knowledge is transferred to Spanish, and a baseline – a model trained *only* on Spanish data – shows the accuracy obtained without any transfer learning. Here, Catalan and Italian help the most, followed by Portuguese, French and, finally, Arabic. This corresponds to the order of lexical similarity with Spanish, except for the performance of Portuguese (cf. Tab. 2). A possible explanation is the potentially confusing overlap of lemmata between the two languages – cf. discussion in the next subsection. That the transfer learning setup improves performance for the unrelated language Arabic as source is at first surprising. However, adding new samples to a small training set helps prevent overfitting (e.g., rote memorization) even if the source is a morphologically unrelated language; effectively acting as a regularizer.

Following (Kann and Schütze, 2016b) we did not use standard regularizers. To verify that the effect of Arabic is mainly a regularization effect, we ran a small monolingual experiment on ES (200 setting) with dropout 0.5 (Srivastava et al., 2014). The resulting accuracy is 0.57, very similar to the comparable Arabic number of 0.54 in the table. The accuracy for dropout and 50 ES samples stays at 0.00, showing that in extreme low-resource settings an unrelated language might be preferable to a standard regularizer.

Error Analysis for Romance. Even for only 50 Spanish instances, many inflections are correctly produced in transfer. For, e.g., (*criar*, 3PlFutSbj) ↦ *criaren*, model outputs are: fr: *criaren*, ca: *criaren*, es: *crntaron*, it: *criaren*, ar: *ecriren*, pt: *criaren* (all correct except for the two baselines). Many errors involve accents, e.g., (*contrastar*, 2PlFutInd) ↦ *contrastaréis*; model outputs are: fr: *contrastareis*, ca: *contrastareis*, es: *conterarían*, it: *contrastareis*, ar: *contastarías*, pt: *contrastareis*. Some inflected forms are produced incorrectly by all systems, mainly because they apply the inflectional rules of the source language directly to the target. Finally, the output of the model trained on Portuguese contains a class of errors that are unlike those of other systems. Example: (*contraatacar*, 1SgCond) ↦ *contraatacaría* with the following solutions: fr: *contratacaríam*, ca: *contraatacaría*, es: *concarnar*, it: *contratacé*, ar: *cuntataría* and pt: *contra-atacaría*. The Portuguese model inserts "-" because Portuguese train data contains *contraatacar* and "-" appears in its inflected form. Thus, it seems that shared lemmata between the high-resource source language and the low-resource target language hurt our model's performance.[4] An

[4]To investigate this in more detail we retrain the Portuguese model with 50 Spanish samples, but exclude all lemmata that appear in Spanish train/dev/test, resulting in only 3695

		PT	CA	IT	CA&PT	CA&IT
		→ES				
50	acc ↑	0.48	**0.58**	0.46	0.56	**0.58**
	ED ↓	0.85	0.80	1.15	**0.67**	0.82
200	acc ↑	0.62	0.78	0.74	0.77	**0.79**
	ED ↓	0.47	0.39	0.44	0.34	**0.31**

Table 4: Results for transfer from pairs of source languages to ES. Results from single languages are repeated for comparison.

		0	PT	CA	IT	FR	AR
		→ES					
one shot	acc ↑	0.00	**0.44**	0.39	0.23	0.13	0.00
	ED ↓	6.26	**1.01**	1.27	1.83	2.87	7.00
zero shot	acc ↑	0.00	**0.14**	0.08	0.01	0.02	0.00
	ED ↓	7.18	**1.95**	1.99	3.12	4.27	7.50

Table 5: Results for one-shot and zero-shot transfer learning. Formatting is the same as for Tab. 3. We still use $n_s = 12000$ source samples. In the one-shot (resp. zero-shot) case, we observe *exactly one form* (resp. *zero forms*) for each tag in the target language at training time.

example for the generally improved performance across languages for 200 Spanish training samples is (*contrastar*, 2PIIndFut) ↦ *contrastaréis*: all models now produce the correct form.

5.2 Exp. 2: Multiple Source Languages

We now want to investigate the effect of multiple source languages.

Data. Our experimental setup is similar to §5.1: we use the same dev, test and low-resource train sets as before. However, we limit this experiment to the Romance language family and the high-resource train data consists of samples from *two different* source languages at once. Choosing those which have the highest accuracies on their own, we experiment with the following pairs: CA&PT, as well as CA&IT. In order to keep all experiments easily comparable, we use *half* of each source language's data, again ending up with a total of 12,000 high-resource samples.

Results and Discussion. Results are shown in Tab. 4. Training on two source languages improves over training on a single one. Increases in accuracy are minor, but edit distance is reduced by up to 0.13 (50 low-resource samples) and 0.08 (200 low-resource samples). That using data from multiple languages is beneficial might be due to a weaker tendency of the final model to adapt wrong rules from the source language, since different alternatives are presented during training.

5.3 Exp. 3: Zero-Shot/One-Shot Transfer

In §5.1, we investigated the relationship between in-domain (target) training set size and performance. Here, we look at the extreme case of training set sizes 1 (one-shot) and 0 (zero-shot) for a tag. We train our model on *a single* sample for *half* of the tags appearing in the low-resource language, i.e., if $\mathcal{T}_\ell$ is the set of morphological tags for the target language, train set size is $|\mathcal{T}_\ell|/2$. As before, we add 12,000 source samples.

We report *one-shot accuracy* (resp. *zero-shot accuracy*), i.e., the accuracy for samples with a tag that has been seen once (resp. never) during training. Note that the model has seen the *individual subtags* each tag is composed of.[5]

Data. Now, we use the same dev, test and high-resource train sets as in §5.1. However, the low-resource data is created in the way specified above. To remove a potentially confounding variable, we impose the condition that no two training samples belong to the same lemma.

Results and Discussion. Tab. 5 shows that the Spanish and Arabic systems do not learn anything useful for either half of the tags. This is not surprising as there is not enough Spanish data for the system to generalize well and Arabic does not contribute exploitable information. The systems trained on French and Italian, in contrast, get a non-zero accuracy for the zero-shot case as well as 0.13 and 0.23, respectively, in the one-shot case. This shows that a single training example is sometimes sufficient for successful generation although generalization to tags never observed is rarely possible. Catalan and Portuguese show the best performance in both settings; this is intuitive since they are the languages closest to the target (cf. Tab. 2). In fact, adding Portuguese to the training data yields an absolute increase in accuracy of 0.44 (0.44 vs. 0.00) for one-shot and 0.14 (0.14 vs. 0.00) for zero-shot with corresponding improvements in edit distance.

Overall, this experiment shows that with transfer learning from a closely related language the per-

training samples. Accuracy on test *increases* by 0.09 despite the reduced size of the training set.

[5] It is very unlikely that due to random selection a subtag will not be in train; this case did not occur in our experiments.

formance of zero-shot morphological generation improves over the monolingual approach, and, in the one-shot setting, it is possible to generate the right form nearly half the time.

5.4 Exp. 4: True Transfer vs. Other Effects

We would like to separate the effects of regularization that we saw for Arabic from true transfer.

To this end, we generate a random cipher (i.e., a function $\gamma : \Sigma \cup \mathcal{S} \mapsto \Sigma \cup \mathcal{S}$) and apply it to all word forms and morphological tags of the high-resource train set; target language data are not changed. Ciphering makes it harder to learn true "linguistic" transfer of morphology. Consider the simplest case of transfer: an identical mapping in two languages, e.g., (*visitar*, 1SgPresInd) $\mapsto$ *visito* in both Portuguese and Spanish. If we transform Portuguese using the cipher γ(iostv...) = kltqa..., then *visito* becomes *aktkql* in Portuguese and its tag becomes similarly unrecognizable as being identical to the Spanish tag 1SgPresInd. Our intuition is that ciphering will disrupt transfer of morphology.[6] On the other hand, the regularization effect we observed with Arabic should still be effective.

Data. We use the Portuguese-Spanish and Arabic-Spanish data from §5.1. We generate a random cipher and apply it to morphological tags and word forms for Portuguese and Arabic. The language tags are kept unchanged. Spanish is also not changed. For comparability with Tab. 3, we use the same dev and test sets as before.

Results and Discussion. Tab. 6 shows that performance of PT→ES drops a lot: from 0.48 to 0.09 for 50 samples and from 0.62 to 0.54 for 200 samples. This is because there are no overt similarities between the two languages left after applying the cipher, e.g., the two previously identical forms *visito* are now different.

The impact of ciphering on AR→ES varies: slightly improved in one case (0.54 vs. 0.56), slightly worse in three cases. We also apply the cipher to the tags and Arabic and Spanish share subtags, e.g., Sg. Just the knowledge that something is a subtag is helpful because subtags must not be generated as part of the output. We can explain the tendency of ciphering to decrease performance on AR→ES by the "masking" of common subtags.

[6]Note that ciphered input is much harder than transfer between two alphabets (Latin/Cyrillic) because it creates ambiguous input. In the example, Spanish "i" is totally different from Portuguese "i" (which is really "k"), but the model must use the same representation.

		0→ES	PT→ES		AR→ES	
			orig	ciph	orig	ciph
50	acc ↑	0.00	0.48	0.09	0.04	0.02
	ED ↓	5.42	0.85	3.25	4.06	4.62
200	acc ↑	0.38	0.62	0.54	0.54	0.56
	ED ↓	1.37	0.57	0.95	0.87	0.93

Table 6: Results for ciphering. "0→ES" and "orig" are original results, copied from Tab. 3; "ciph" is the result after the cipher has been applied.

For 200 samples and ciphering, there is no clear difference in performance between Portuguese and Arabic. However, for 50 samples and ciphering, Portuguese (0.09) seems to perform better than Arabic (0.02) in accuracy. Portuguese uses suffixation for inflection whereas Arabic is templatic and inflectional changes are not limited to the end of the word. This difference is not affected by ciphering. Perhaps even ciphered Portugese lets the model learn better that the beginnings of words just need to be copied. For 200 samples, the Spanish dataset may be large enough, so that ciphered Portuguese no longer helps in this regard.

Comparing no transfer with transfer from a ciphered language to Spanish, we see large performance gains, at least for the 200 sample case: 0.38 (0→ES) vs. 0.54 (PT→ES) and 0.56 (AR→ES). This is evidence that our conjecture is correct that the baseline Arabic mainly acts as a regularizer that prevents the model from memorizing the training set and therefore improves performance. So performance improves even though no true transfer of morphological knowledge takes place.

6 Related Work

Cross-lingual transfer learning has been used for many tasks, e.g., automatic speech recognition (Huang et al., 2013), parsing (Cohen et al., 2011; Søgaard, 2011; Naseem et al., 2012; Ammar et al., 2016), language modeling (Tsvetkov et al., 2016), entity recognition (Wang and Manning, 2014b) and machine translation (Johnson et al., 2016; Ha et al., 2016).

One straightforward method is to translate datasets and then train a monolingual model (Fortuna and Shawe-Taylor, 2005; Olsson et al., 2005). Also, aligned corpora have been used to project information from annotations in one language to another (Yarowsky et al., 2001; Padó and Lapata, 2005). The drawback is that machine translation

errors cause errors in the target. Therefore, alternative methods have been proposed, e.g., to port a model trained on the source language to the target language (Shi et al., 2010).

In the realm of morphology, Buys and Botha (2016) recently adapted methods for the training of POS taggers to learn weakly supervised morphological taggers with the help of parallel text. Snyder and Barzilay (2008a, 2008b) developed a non-parametric Bayesian model for morphological segmentation. They performed identification of cross-lingual abstract morphemes and segmentation simultaneously and reported, similar to us, best results for related languages.

Work on **paradigm completion** has recently been encouraged by the SIGMORPHON 2016 shared task on morphological reinflection (Cotterell et al., 2016a). Some work first applies an unsupervised alignment model to source and target string pairs and then learns a string-to-string mapping (Durrett and DeNero, 2013; Nicolai et al., 2015), using, e.g., a semi-Markov conditional random field (Sarawagi and Cohen, 2004). Encoder-decoder RNNs (Aharoni et al., 2016; Faruqui et al., 2016; Kann and Schütze, 2016b), a method which our work further develops for the cross-lingual scenario, define the current state of the art.

Encoder-decoder RNNs were developed in parallel by Cho et al. (2014) and Sutskever et al. (2014) for machine translation and extended by Bahdanau et al. (2015) with an attention mechanism, supporting better generalization. They have been applied to NLP tasks like speech recognition (Graves and Schmidhuber, 2005; Graves et al., 2013), parsing (Vinyals et al., 2015) and segmentation (Kann et al., 2016).

More recently, a number of papers have used encoder-decoder RNNs in *multitask and transfer learning settings*; this is mainly work in machine translation: (Dong et al., 2015; Zoph and Knight, 2016; Chu et al., 2017; Johnson et al., 2016; Luong et al., 2016; Firat et al., 2016; Ha et al., 2016), *inter alia*. Each of these papers has both similarities and differences with our approach. (i) Most train several distinct models whereas we train a *single model* on input augmented with an explicit encoding of the language (similar to (Johnson et al., 2016)). (ii) Let k and m be the number of different input and output languages. We address the case $k \in \{1, 2, 3\}$ and $m = k$. Other work has addressed cases with $k > 3$ or $m > 3$; this would be an interesting avenue of future research for paradigm completion. (iii) Whereas training RNNs in machine translation is hard, we only experienced one difficult issue in our experiments (due to the low-resource setting): regularization. (iv) Some work is word- or subword-based, our work is character-based. The same way that similar word embeddings are learned for the inputs *cow* and *vache* (French for "cow") in machine translation, we expect similar embeddings to be learned for similar Cyrillic/Latin characters. (v) Similar to work in machine translation, we show that zero-shot (and, by extension, one-shot) learning is possible.

(Ha et al., 2016) (which was developed in parallel to our transfer model although we did not prepublish our paper on arxiv) is most similar to our work. Whereas Ha et al. (2016) address machine translation, we focus on the task of paradigm completion in low-resource settings and establish the state of the art for this problem.

7 Conclusion

We presented a cross-lingual transfer learning method for paradigm completion, based on an RNN encoder-decoder model. Our experiments showed that information from a high-resource language can be leveraged for paradigm completion in a related low-resource language. Our analysis indicated that the degree to which the source language data helps for a certain target language depends on their relatedness. Our method led to significant improvements in settings with limited training data – up to 58% absolute improvement in accuracy – and, thus, enables the use of state-of-the-art models for paradigm completion in low-resource languages.

8 Future Work

In the future, we want to develop methods to make better use of languages with different alphabets or morphosyntactic features, in order to increase the applicability of our knowledge transfer method.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. We are grateful to Siemens and Volkswagenstiftung for their generous support. This research would not have been possible without the organizers of the SIGMORPHON shared task, especially John Sylak-Glassman and Christo Kirov, who created the resources we use.

References

Daniel Abondolo. 2015. *The Uralic Languages*. Routledge.

Roee Aharoni, Yoav Goldberg, and Yonatan Belinkov. 2016. Improving sequence to sequence learning for morphological inflection generation: The BIU-MIT systems for the SIGMORPHON 2016 shared task for morphological reinflection. In *SIGMORPHON*.

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

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *TACL* 4:431–444.

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

Jan Buys and Jan A Botha. 2016. Cross-lingual morphological tagging for low-resource languages. In *ACL*.

Rich Caruana. 1997. Multitask learning. *Machine Learning* 28(1):41–75.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint 1409.1259* .

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of simple domain adaptation methods for neural machine translation. *arXiv preprint 1701.03214* .

Shay B Cohen, Dipanjan Das, and Noah A Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *EMNLP*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR* 12(Aug):2493–2537.

Greville Corbett and Bernard Comrie. 2003. *The Slavonic Languages*. Routledge.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016a. The SIGMORPHON 2016 shared task—morphological reinflection. In *SIGMORPHON*.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2015. Modeling word forms using latent underlying morphs and phonology. *TACL* 3:433–447.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016b. Morphological smoothing and extrapolation of word embeddings. In *ACL*.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. 2007. Analysis of morph-based speech recognition and the modeling of out-of-vocabulary words across languages. In *NAACL-HLT*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *ACL-IJCNLP*.

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

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *ACL*.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *NAACL*.

Orhan Firat, KyungHyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *CoRR* abs/1601.01073.

Blaz Fortuna and John Shawe-Taylor. 2005. The use of machine translation tools for cross-lingual text mining. In *ICML Workshop on Learning with Multiple Views*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *IEEE*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18(5):602–610.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint 1611.04798* .

Martin Harris and Nigel Vincent. 2003. *The Romance languages*. Routledge.

Robert Hetzron. 2013. *The Semitic Languages*. Routledge.

Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and n Gong. 2013. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *IEEE*.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR* abs/1611.04558.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural morphological analysis: Encoding-decoding canonical segments. In *EMNLP*.

Katharina Kann and Hinrich Schütze. 2016a. Single-model encoder-decoder with explicit morphological representation for reinflection. In *ACL*.

Katharina Kann and Hinrich Schütze. 2016b. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *ACL*.

Christo Kirov, John Sylak-Glassman, Roger Que, and David Yarowsky. 2016. Very-large scale parsing and normalization of wiktionary morphological paradigms. In *LREC*.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *CoRR* abs/1504.00941.

M Paul Lewis, editor. 2009. *Ethnologue: Languages of the World*. SIL International, Dallas, Texas, 16 edition.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*.

Karthik Narasimhan, Damianos Karakos, Richard Schwartz, Stavros Tsakalidis, and Regina Barzilay. 2014. Morphological segmentation for keyword spotting. In *EMNLP*.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *ACL*.

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

J Scott Olsson, Douglas W Oard, and Jan Hajič. 2005. Cross-language text classification. In *ACM SIGIR*.

Sebastian Padó and Mirella Lapata. 2005. Cross-linguistic projection of role-semantic information. In *HLT/EMNLP*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *LREC*.

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

Wolfgang Seeker and Özlem Çetinoğlu. 2015. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *TACL* 3:359–373.

Lei Shi, Rada Mihalcea, and Mingjun Tian. 2010. Cross language text classification by model translation and semi-supervised learning. In *EMNLP*.

Benjamin Snyder and Regina Barzilay. 2008a. Cross-lingual propagation for morphological analysis. In *AAAI*.

Benjamin Snyder and Regina Barzilay. 2008b. Unsupervised multilingual learning for morphological segmentation. In *ACL-HLT*.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *ACL-HLT*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

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

John Sylak-Glassman. 2016. The composition and use of the universal morphological feature schema (unimorph schema). Technical report, Department of Computer Science, Johns Hopkins University.

John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015. A language-independent feature schema for inflectional morphology. In *ACL-IJCNLP*.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *NAACL-HLT*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *NIPS*.

Mengqiu Wang and Christopher D Manning. 2014a. Cross-lingual projected expectation regularization for weakly supervised learning. *TACL* 2:55–66.

Mengqiu Wang and Christopher D Manning. 2014b. Cross-lingual pseudo-projected expectation regularization for weakly supervised learning. *TACL* 2:55–66.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *EMNLP*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *HLT*.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *NAACL-HLT*.

Chapter 5

Unlabeled Data for Morphological Generation With Character-Based Sequence-to-Sequence Models

Unlabeled Data for Morphological Generation With Character-Based Sequence-to-Sequence Models

Katharina Kann and **Hinrich Schütze**
LMU Munich, Germany
kann@cis.lmu.de

Abstract

We present a semi-supervised way of training a character-based encoder-decoder recurrent neural network for morphological reinflection, the task of generating one inflected word form from another. This is achieved by using unlabeled tokens or random strings as training data for an autoencoding task, adapting a network for morphological reinflection, and performing multi-task training. We thus use limited labeled data more effectively, obtaining up to 9.9% improvement over state-of-the-art baselines for 8 different languages.

1 Introduction

Morphologically rich languages use inflection—the adaptation of a surface form to its syntactic context—to mark the properties of a word, e.g., *gender* or *number* of nouns or *tense* of verbs. This drastically increases the type-token ratio, and thus negatively effects natural language processing (NLP), making morphological analysis and generation an important field of research.

In this work, we focus on morphological reinflection (MRI), the task of mapping one inflected form of a lemma to another, given the morphological properties of the target, e.g., (*smiling*, *PastPart*) → *smiled*. The lemma does not have to be known. Recently, there have been some advances on the topic, motivated by the SIGMORPHON 2016 shared task on morphological reinflection (Cotterell et al., 2016) and the CoNLL-SIGMORPHON 2017 shared task on universal morphological reinflection (Cotterell et al., 2017). In 2016, neural sequence-to-sequence models, specifically attention-based encoder-decoder models, outperformed all other approaches by a wide margin (Faruqui et al., 2016; Kann and Schütze, 2016). However, those models require a lot of training data, while in contrast many morphologically rich languages are low-resource, and little work has been done so far on neural models for morphology in settings with limited training data. This makes sequence-to-sequence models not applicable to morphological generation in most languages.

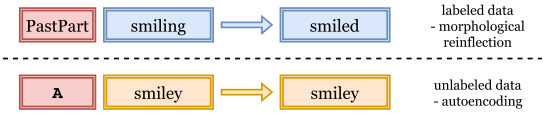


Figure 1: Examples for labeled and unlabeled input. The content of the red boxes (very left in both rows) signalizes if the sample belongs to the MRI task or the autoencoding task.

An abundance of *unlabeled* data, in contrast, can be assumed available for each language in the focus of NLP. Thus, we propose a semi-supervised training method for a state-of-the-art encoder-decoder network for MRI using both labeled and unlabeled data, mitigating the need for time-expensive annotations. We achieve this by treating unlabeled words as training examples for an *autoencoding* (Vincent et al., 2010) task and multi-task training (cf. Figure 1). We intuit the following reasons why this should be beneficial: (i) The decoder's character language model can be trained using unlabeled data. (ii) Training on a second task reduces the problem of overfitting. (iii) By forcing the model to additionally learn autoencoding, we give it a strong prior to copy the input string. This might be advantageous as often many forms of a paradigm share the same stem, e.g., ***smil**ing* and ***smil**ed*. In order to investigate the importance of the latter, we further experiment with autoencoding of *random strings* and find that for our experimental settings and non-templatic languages the performance gain is comparable to using corpus words.

2 Model Description

The log-likelihood for joint training on the tasks of MRI and autoencoding is:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(f_s, f_t, t) \in \mathcal{T}} \log p_{\boldsymbol{\theta}}\left(f_t \mid e(f_s, t)\right) + \sum_{w \in \mathcal{W}} \log p_{\boldsymbol{\theta}}(w \mid e(w)), \quad (1)$$

$\mathcal{T}$ is the MRI training data, with each example consisting of a source form f_s, a target form f_t and a target tag t. $\mathcal{W}$ denotes a set of words in the language of the system. The encoding function e depends on $\boldsymbol{\theta}$. The parameters $\boldsymbol{\theta}$ are shared across the two tasks, resulting in a share of information. We obtain this by giving our model data from both sets at the same time, and marking each example with a task-specific input symbol, cf. Figure 1. Following (Kann and Schütze, 2016), we employ a neural encoder-decoder model.

Encoder. For the input of the encoder, we adapt the format by Kann and Schütze (2016), but modify it to be able to handle unlabeled data: Given the set of morphological subtags M each target tag is composed of (e.g., the tag *1SgPresInd* contains the subtags *1*, *Sg*, *Pres* and *Ind*), and the alphabet Σ of the language of application, our input is of the form $\texttt{B}[\mathbf{A}/\text{M}^*]\Sigma^*\texttt{E}$, i.e., it consists of *either* a sequence of subtags *or* the symbol **A** signaling that the input is not annotated and should be autoencoded, and (in both cases) the character sequence of the input word. B and E are start and end symbols. Each part of the input is represented by an embedding.

We then encode the input $x = x_1, x_2, \ldots, x_{T_x}$ using a bidirectional gated recurrent neural network (GRU) (Cho et al., 2014b), i.e., $\overrightarrow{h}_i = f\left(\overrightarrow{h}_{i-1}, x_i\right)$ and $\overleftarrow{h}_i = f\left(\overleftarrow{h}_{i+1}, x_i\right)$, with f being the update function of the hidden layer. Forward and backward hidden states are concatenated to obtain the input h_i for the decoder.

Decoder. The decoder is an attention-based GRU, defining a probability distribution over strings in Σ^*:

$$p(y \mid x) = \prod_{t=1}^{T_y} p(y_t \mid y_1, \ldots, y_{t-1}, s_t, c_t),$$

with s_t being the decoder hidden state for time t and c_t being a context vector, calculated using the encoder hidden states together with attention weights. A detailed description of the model can be found in Bahdanau et al. (2015).

3 Experiments

Dataset. We experiment on the task 3 dataset of the SIGMORPHON 2016 shared task on MRI (Cotterell et al., 2016) and all standard languages provided: Arabic, Finnish, Georgian, German, Navajo, Russian, Spanish and Turkish. German, Spanish and Russian are suffixing and exhibit stem changes. Russian differs from the other two in that those stem changes are consonantal and not vocalic. Finnish and Turkish are agglutinating, almost exclusively suffixing and have vowel harmony systems. Georgian uses both prefixiation and suffixiation. In contrast, Navajo mainly makes use of prefixes with consonant harmony among its sibilants. Finally, Arabic is a templatic, non-concatenative language.

For each language, we further add randomly sampled words from the respective Wikipedia dumps. We exclude tokens that are not exclusively composed from characters of the language's alphabet, e.g., digits, or do not appear at least 2 times in the corpus. The exact amount of unlabaled data added is treated as a hyperparameter depending on the number of available annotated examples and optimized on the development set, cf. Section 4.1. Evaluation is done on the official shared task test set.

Training, hyperparameters and evaluation. We mainly adopt the hyperparameters of (Kann and Schütze, 2016). Embeddings are 300-dimensional, the size of all hidden layers is 100 and for training we use ADADELTA (Zeiler, 2012) with a batch size of 20. We train all models which use $\frac{1}{8}$ or more of the labeled data for 200 epochs, and models that see $\frac{1}{16}$ and $\frac{1}{32}$ of the original data for 400 and 800 epochs, respectively. In all cases, we apply the last model for testing.

We evaluate using two metrics: accuracy and edit distance. Accuracy reports the percentage of completely correct solutions, while the edit distance between the system's guess and the gold solution gives credit to systems that produce forms that are close to the right form.

Baselines. We compare our system to three baselines: The first one is **MED**[1], the winning sys-

[1] http://cistern.cis.lmu.de/med/

		ar				fi				ka				de				nv				ru				sp				tu			
		SIG16	SIG17	MED	Our	SIG16	SIG17	MED	Our	SIG16	SIG17	MED	Our	SIG16	SIG17	MED	Our	SIG16	SIG17	MED	Our	SIG16	SIG17	MED	Our	SIG16	SIG17	MED	Our	SIG16	SIG17	MED	Our
$\frac{1}{4}$	acc	.188	.094	.716	**.722**	.293	.325	.809	**.854**	.814	.831	.910	**.912**	.721	.687	.882	**.888**	.317	.403	.706	**.711**	.641	.638	**.825**	.824	.558	.539	.939	**.942**	.181	.129	.904	**.910**
	ED	2.26	3.06	0.94	**0.92**	1.90	1.47	0.47	**0.35**	0.42	0.38	**0.28**	0.30	0.47	0.54	0.33	**0.31**	2.04	1.95	1.01	**0.97**	0.69	0.65	**0.43**	**0.43**	0.96	0.97	**0.15**	**0.15**	2.92	3.33	0.27	**0.23**
$\frac{1}{8}$	acc	.104	.063	.600	**.640**	.207	.227	.687	**.732**	.798	.791	.883	**.894**	.618	.593	.851	**.873**	.247	.350	.516	**.619**	.516	.523	.766	**.772**	.441	.409	.896	**.916**	.120	.080	**.846**	.832
	ED	2.76	3.32	1.37	**1.20**	2.32	1.91	0.85	**0.77**	0.47	0.44	0.45	**0.42**	0.67	0.73	0.42	**0.35**	2.40	2.23	1.75	**1.40**	0.95	0.92	**0.60**	**0.60**	1.36	1.35	0.26	**0.22**	3.42	3.80	**0.47**	0.54
$\frac{1}{16}$	acc	.052	.043	.470	**.533**	.126	.149	.543	**.620**	.709	.751	.860	**.875**	.504	.495	.791	**.839**	.204	.329	.350	**.473**	.384	.422	.645	**.695**	.317	.308	.807	**.862**	.070	.049	.717	**.739**
	ED	3.36	3.53	1.80	**1.59**	2.84	2.34	1.33	**1.16**	0.62	**0.50**	0.58	0.52	0.90	0.94	0.60	**0.45**	2.71	2.41	2.63	**2.05**	1.23	1.17	0.94	**0.82**	1.80	1.70	0.47	**0.36**	3.81	4.09	0.99	**0.94**
$\frac{1}{32}$	acc	.028	.027	.263	**.381**	.073	.088	.314	**.402**	.595	.648	.818	**.852**	.384	.386	.661	**.722**	.174	.303	.174	**.369**	.249	.293	.406	**.502**	.196	.245	.657	**.756**	.044	.028	.524	**.571**
	ED	3.73	3.73	2.79	**2.22**	3.18	2.76	2.48	**2.00**	0.87	0.70	0.76	**0.65**	1.15	1.18	1.01	**0.90**	2.94	**2.65**	3.85	2.73	1.61	1.45	1.71	**1.38**	2.22	2.06	0.97	**0.62**	4.19	4.27	1.98	**1.80**

Table 1: Accuracy (the higher the better) and edit distance (the lower the better) for our system and the three baselines on the official test set of task 3 of the SIGMORPHON 2016 shared task. Only the indicated amount (row labels) of the original training data is used, emulating a low-resource setting. Best results for each language in bold.

tem of the 2016 shared task. The network architecture is the same as in our system, but it is trained exclusively on labeled data. Thus, we expect it to suffer stronger from a lack of resources.

The second baseline is the official SIGMORPHON 2016 shared task baseline (**SIG16**) (Cotterell et al., 2016), which is similar in spirit to the system described by Nicolai et al. (2015). The system treats the prediction of edit operations to be performed on the input string as a sequential decision-making problem, greedily choosing each edit action given the previously chosen actions. The selection of operations is made by an averaged perceptron, using the binary features described in (Cotterell et al., 2016).[2]

Third, we compare to the baseline system of the CoNLL-SIGMORPHON 2017 shared task on universal morphological reinflection (**SIG17**) (Cotterell et al., 2017), which is extremely suitable for low-resource settings. It splits all source and target forms in the training set into prefix, middle part and suffix, and uses those to find prefix or suffix substitution rules. Every evaluation example is searched for the longest contained prefix or suffix and the rule belonging to the affix and given target tag is applied to obtain the output.

Results and discussion. As shown in Table 1, additionally training on unlabeled examples improves the performance of the encoder-decoder network for nearly all settings and languages, especially for the very low-resource scenarios with $\frac{1}{16}$ and $\frac{1}{32}$ of the training data. The biggest increase in accuracy can be seen for Russian and Spanish, both in the $\frac{1}{32}$ setting, with 0.0963 $(0.5023 - 0.4060)$ and 0.0992 $(0.7564 - 0.6572)$, respectively. For the settings with bigger amounts of training data available, the unlabeled data does not change performance a lot. This was expected, as the model already gets enough information from the annotated data. However, semi-supervised training never *hurts* performance, and can thus always be employed. Overall, our semi-supervised training method shows to be a useful extension of the original system.

Furthermore, there are only two cases—Georgian, $\frac{1}{16}$, and Navajo, $\frac{1}{32}$—where any of the SIGMORPHON baselines outperforms the neural methods. This clearly shows the superiority of neural networks for the task and emphasizes the need to reduce the amount of labeled training data required for their training.

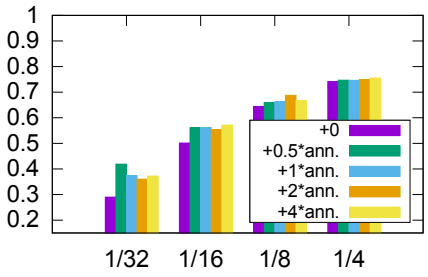
4 Analyses

4.1 Amount of Unlabeled Data

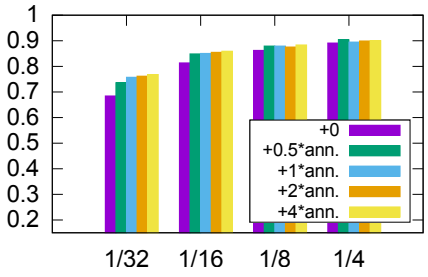
We now consider the amount of unlabeled examples as a function of the number of annotated examples. Data and training regime are the same as in Section 3. This analysis is performed on the development set and we report the highest accuracy obtained during training.

The resulting accuracies for Arabic and German can be seen in Figure 2. The other languages behave similarly to German. The loss of performance for reducing the training data varies a lot between languages, depending on how regular and thus "easy to learn" those are. Concerning the amount of unlabeled examples, it seems that even though in single cases other ratios are slightly better, using 4 times more unlabeled examples mostly obtains highest accuracy. Thus, a general rule could be that the more additional examples are used the better. The only exception is Arabic in the $\frac{1}{32}$ setting, where using half as many unlabeled as labeled examples obtains much better results. We explain this with the Semitic language being templatic. Since words in Arabic paradigms do

[2]Note that our use of the system differs from the official baseline in that we perform a direct form-to-form mapping. The shared task system predicts first form-to-lemma and then lemma-to-form. However, we assume no lemmata to be given, and thus are unable to train such a system.



Arabic



German

Figure 2: Comparison of different amounts of unlabeled data, sorted by the amount of labeled training examples in portions of the original data. Evaluated on the development set.

not share a connected stem, we expect that giving the model too much bias to copy might be harming performance in low-resource settings. However, even for low-resource Arabic, using a ratio of 1:4 of labeled to unlabeled examples still yields a better performance than not using unlabeled examples at all. Thus, we can conclude that if aiming for a language-independent setup, this is a good ratio.

4.2 Autoencoding of Random Strings

We expect the network to benefit from a bias to copy strings. This suggests that *any* random combination of characters from the language's alphabet could be autoencoded in order to improve the performance in low-resource settings. To verify this, we train models on new datasets with $\frac{1}{32}$ of the labeled examples from task 3 of the SIGMORPHON 2016 shared task and the optimal number of unlabeled examples for each language, cf. §4.1. However, the unlabeled examples are now random strings of a length between 3 and 20. All models are trained as before. Accuracies on the official test sets are shown in Table 2, and compared to (i) training without unlabeled examples and (ii) the data being enhanced by corpus words. Several aspects of the results are eye-catching. First, for Arabic, the gap to the performance with corpus words is the biggest, showing that indeed the tendency of languages to copy the stem when inflecting is playing an important role. Second, for some languages the performance gains for corpus words and random words are comparable. Third, the performance of random strings is closer to the performance of corpus words the higher the overall accuracy is. The additional unlabeled examples might be acting as regularizers in this case.

	ar	**fi**	**ka**	**de**	**nv**	**ru**	**es**	**tu**
MED	.2628	.3144	.8184	.6608	.1738	.4060	.6572	.5238
MED+*corpus*	**.3811**	**.4015**	.8523	.7221	**.3688**	**.5023**	.7564	**.5713**
MED+*random*	.3064	.3793	**.8531**	**.7313**	.3250	.4958	**.7676**	.5706

Table 2: Accuracies for MED (Kann and Schütze (2016)), MED+*corpus* and MED+*random*. Descriptions in the text.

Overall, this experiment shows clearly that giving the model a bias to copy strings helps for inflection in non-templatic languages, and that random strings can improve a network for MRI.

5 Related Work

For the SIGMORPHON 2016 and the CoNLL-SIGMORPHON 2017 shared tasks (Cotterell et al., 2016, 2017), multiple MRI systems were developed, e.g., (Nicolai et al., 2016; Taji et al., 2016; Kann and Schütze, 2016; Aharoni et al., 2016; Östling, 2016; Makarov et al., 2017). Encoder-decoder neural networks (Cho et al., 2014a; Sutskever et al., 2014; Bahdanau et al., 2015) performed best, such that we extend them in this work. Earlier work on paradigm completion included (Faruqui et al., 2016; Nicolai et al., 2015; Durrett and DeNero, 2013). Work directly tackling MRI was more rare, e.g., (Dreyer and Eisner, 2009). Our work relates to the line of research on minimally supervised and unsupervised methods for morphology, e.g., Creutz and Lagus (2007) and Goldsmith (2001) presenting the unsupervised morphological segmentation systems Morfessor and Linguistica, or (Dreyer and Eisner, 2011; Poon et al., 2009; Snyder and Barzilay, 2008). However, none of those focused directly on MRI or on training neural networks for morphology. The only case we know of where this *was* done was work by Kann et al. (2017). They leveraged morphologically annotated data in a closely related high-resource language to reduce the need for labeled data in the target language. This works well for similar languages, but has the shortcoming to require annotations in such a language to be at hand. A similar approach was presented

by Ha et al. (2016) for machine translation (MT). Unlabeled corpora were used for semi-supervised training of models for MT, e.g., by Cheng et al. (2016); Vincent et al. (2010); Socher et al. (2011); Ramachandran et al. (2016). Those approaches differ from ours, due to a fundamental difference between the two tasks: For MRI, the source vocabulary and the target vocabulary are mostly the same. This makes it intuitive for MRI to train the final model jointly on MRI and autoencoding.

6 Conclusion

We presented a way of semi-supervised training of a state-of-the-art model for low-resource MRI, using words from an unlabeled corpus. We found that the best ratio of labeled to unlabeled data depends of the morphological typology of the language. Finally, we showed that autoencoding random strings also increases performance, for some languages as much as using corpus words.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work was supported by DFG (SCHU2246/10).

References

Roee Aharoni, Yoav Goldberg, and Yonatan Belinkov. 2016. Improving sequence to sequence learning for morphological inflection generation: The BIU-MIT systems for the SIGMORPHON 2016 shared task for morphological reinflection. In *SIGMORPHON*.

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

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. *arXiv preprint arXiv:1606.04596* .

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *SSST*.

Kyunghyun Cho, Bart Van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMNLP*.

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. The CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *CoNLL-SIGMORPHON*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *SIGMORPHON*.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *TSLP* 4(1):3.

Markus Dreyer and Jason Eisner. 2009. Graphical models over multiple strings. In *EMNLP*.

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

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

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *NAACL*.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational linguistics* 27(2):153–198.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798* .

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. One-shot neural cross-lingual transfer for paradig completion. In *ACL*.

Katharina Kann and Hinrich Schütze. 2016. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *ACL*.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and copy: UZH at SIGMORPHON 2017 shared task for morphological reinflection. In *CoNLL-SIGMORPHON*.

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

Garrett Nicolai, Bradley Hauer, Adam St Arnaud, and Grzegorz Kondrak. 2016. Morphological reinflection via discriminative string transduction. In *SIGMORPHON*.

Robert Östling. 2016. Morphological reinflection with convolutional neural networks. In *SIGMORPHON*.

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

Prajit Ramachandran, Peter J Liu, and Quoc V Le. 2016. Unsupervised pretraining for sequence to sequence learning. *arXiv preprint arXiv:1611.02683* .

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *ACL*.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*.

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

Dima Taji, Ramy Eskander, Nizar Habash, and Owen Rambow. 2016. The Columbia University - New York University Abu Dhabi SIGMORPHON 2016 morphological reinflection shared task submission. In *SIGMORPHON*.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 11(Dec):3371–3408.

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

Chapter 6

Neural Multi-Source Morphological Reinflection

Neural Multi-Source Morphological Reinflection

Katharina Kann
CIS
LMU Munich, Germany
kann@cis.lmu.de

Ryan Cotterell
Department of Computer Science
Johns Hopkins University, USA
ryan.cotterell@jhu.edu

Hinrich Schütze
CIS
LMU Munich, Germany
inquiries@cislmu.org

Abstract

We explore the task of multi-source morphological reinflection, which generalizes the standard, single-source version. The input consists of (i) a target tag and (ii) multiple pairs of source form and source tag for a lemma. The motivation is that it is beneficial to have access to more than one source form since different source forms can provide complementary information, e.g., different stems. We further present a novel extension to the encoder-decoder recurrent neural architecture, consisting of multiple encoders, to better solve the task. We show that our new architecture outperforms single-source reinflection models and publish our dataset for multi-source morphological reinflection to facilitate future research.

1 Introduction

Morphologically rich languages still constitute a challenge for natural language processing (NLP). The increased data sparsity caused by highly inflected word forms in certain languages causes otherwise state-of-the-art systems to perform worse in standard tasks, e.g., parsing (Ballesteros et al., 2015) and machine translation (Bojar et al., 2016). To create systems whose performance is not deterred by complex morphology, the development of NLP tools for the generation and analysis of morphological forms is crucial. Indeed, these considerations have motivated a great deal of recent work on the topic (Ahlberg et al., 2015; Dreyer, 2011; Nicolai et al., 2015).

In the area of generation, the most natural task is morphological inflection—finding an inflected form for a given target tag and lemma. An example for English is as follows: (`trg:3rdSgPres`, *bring*) $\mapsto$ *brings*. In this case, the 3rd person singular present tense of *bring* is generated. One generalization of inflection is morphological *re*inflection (MRI) (Cotterell et al., 2016), where we must produce an inflected form from a triple of target tag, source form and source tag. The inflection task is the special case where the source form is the lemma. As an example, we may again consider generating the English past tense form from the 3rd person singular present: (`trg:3rdSgPres`, *brought*, `src:Past`) $\mapsto$ *brings* (where `trg` = "target tag" and `src` = "source tag"). As the starting point varies, MRI is more difficult than morphological inflection and exhibits more data sparsity. However, it is also more widely applicable since lexical resources are not always complete and, thus, the lemma is not always available. A more complex German example is given in Table 1.

	Present Ind		Past Ind		Past Sbj	
	Sg	Pl	Sg	Pl	Sg	Pl
1	treffe	treffen	traf	trafen	träfe	träfen
2	triffst	trefft	trafst	traft	träfest	träfet
3	trifft	treffen	traf	trafen	träfe	träfen

Table 1: The paradigm of the strong German verb TREFFEN, which exhibits an irregular ablaut pattern. Different parts of the paradigm make use of one of four bolded theme vowels: e, i, a or ä. In a sense, the verbal paradigm is partitioned into subparadigms. To see why multi-source models could help in this case, starting only from the infinitive treffen makes it difficult to predict subjunctive form träfest, but the additional information of the fellow subjunctive form träfe makes the task easier.

In this work, we generalize the MRI task to a multi-source setup. Instead of using a single source form-tag pair, we use *multiple* source form-tag pairs. Our motivation is that (i) it is often beneficial to have access to more than one source form since different source forms can provide complementary information, e.g., different stems; and (ii)

in many application scenarios, we will have encountered more than one form of a paradigm at the point when we want to generate a new form.

We will make the intuition that multiple source forms provide complementary information precise in the next section, but first return to the English verb *bring*. Generating the form *brings* from *brought* may be tricky—there is an irregular vowel shift. However, if we had a second form with the same theme vowel, e.g., *bringing*, the task would be much easier, i.e., (`trg`:3rdSgPres, `form1`:*brought*, `src1`:Past, `form2`:*bringing*, `src2`:Gerund). A multi-source approach clearly is advantageous for this case since mapping *bringing* to *brings* is regular even though the verb itself is irregular.

The contributions of the paper are as follows. (i) We define the task of multi-source MRI, a generalization of single-source MRI. (ii) We show that a multi-source MRI system, implemented as a novel encoder-decoder, outperforms the top-performing system in the SIGMORPHON 2016 Shared Task on Morphological Reinflection on seven out of eight languages, when given additional source forms. (iii) We release our data to support the development of new systems for MRI.

2 The Task: Multi-Source Reinflection

Previous work on morphological reinflection has assumed a single source form, i.e., an input consisting of exactly one inflected source form (potentially the lemma) and the corresponding morphological tag. The output is generated from this input. In contrast, multi-source morphological reinflection, the task we introduce, is a generalization in which the model receives multiple form-tag pairs. In effect, this gives the model a partially annotated paradigm from which it predicts the rest.

The multi-source variant is a more natural problem than single-source morphological reinflection since we often have access to more than just one form.[1] For example, corpora such as the universal dependency corpus (McDonald et al., 2013) that are annotated on the token level with inflectional features often contain several different inflected forms of a lemma. Such corpora would provide an ideal source of data for the multi-source MRI task.

Formally, we can think of a morphological paradigm as follows. Let Σ be a discrete alphabet for a given language and $\mathcal{T}$ be the set of morphological tags in the language. The inflectional table or morphological paradigm π of a lemma w can be formalized as a set of pairs:

$$\pi(w) = \{(f_1, t_1), (f_2, t_2), \ldots, (f_N, t_N)\}, \quad (1)$$

where $f_i \in \Sigma^+$ is an inflected form of w, and $t_i \in \mathcal{T}$ is the morphological tag of the form f_i. The integer N is the number of slots in the paradigm that have the syntactic category (POS) of w.

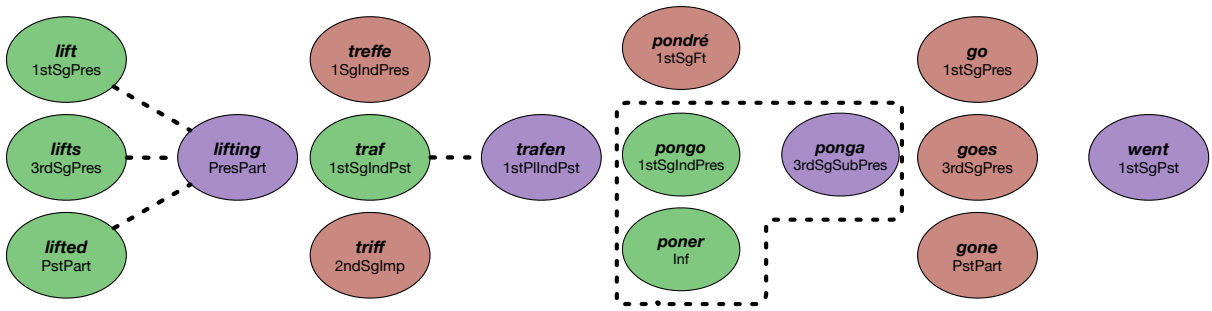
Using this notation, single-source morphological reinflection (MRI) can be described as follows. Given a target tag and a pair of source form and source tag $(t_{\text{trg}}, (f_{\text{src}}, t_{\text{src}}))$ as input, predict the target form f_{trg}. There has been a substantial amount of prior work on this task, including systems that participated in Task 2 of the SIGMORPHON 2016 shared task (Cotterell et al., 2016). Thus, we may define the task of *multi-source morphological reinflection* as follows: Given a target tag and a set of k form-tag source pairs $(t_{\text{trg}}, \{(f^1_{\text{src}}, t^1_{\text{src}}), \ldots, (f^k_{\text{src}}, t^k_{\text{src}})\})$ as input, predict the target form f_{trg}. Note that single-source MRI is a special case of multi-source MRI for $k = 1$.

2.1 Motivating Examples

Figure 1 gives examples for four different configurations that can occur in multi-source MRI.[2] We have colored the source forms green and drawn a dotted line to the target if they contain sufficient information for correct generation. If two source forms together are needed, the dotted line encloses both of them. Source forms that provide no information in the configuration are colored red (no arrow); note these forms could provide (and in most

[1] Scenarios where a single form is available and that form is the lemma are perhaps not infrequent. In high-resource languages, an electronic dictionary may have near-complete coverage of the lemmata of the language. However, paradigm completion is especially crucial for neologisms and low-resource languages.

[2] Figure 1 is not intended as a complete taxonomy of possible MRI configurations, e.g., there are hybrids of ANYFORM and NOFORM (some forms are informative, others are suppletive) and fuzzy variants (a single form gives pretty good evidence for how to generate the target form, but another single form gives better evidence). All of our examples make additional assumptions, e.g., that we have not seen other similar forms in training either of the same lemma (e.g., *poner*) or of a similar lemma (e.g., *reponer*). Hopefully, the examples are illustrative of the main conceptual distinction: several single forms each are sufficient by themselves (ANYFORM), a single, but carefully selected form is sufficient (SINGLEFORM), multiple forms are needed to generate the target (MULTIFORM) and the target form cannot be predicted (irregular) from the source forms (NOFORM).



(a) ANYFORM (b) SINGLEFORM (c) MULTIFORM (d) NOFORM

Figure 1: Four possible input configurations in multi-source morphological reinflection (MRI). In each subfigure, the target form on the right is **purple**. The source forms are on the left and are **green** if they can be used to predict the target form (also connected with a dotted line) and **red** if they cannot. There are four possible configurations: (i) ANYFORM is the case where one can predict the target form from any of the source forms. (ii) SINGLEFORM is the case where only one form can be used to regularly predict the target form. (iii) MULTIFORM is the case where multiple forms are *necessary* to predict the target form. (iv) NOFORM is the case where the target form cannot be regularly derived from any of the source forms. Multi-source MRI is expected to perform better than single-source MRI for the configurations SINGLEFORM and MULTIFORM, but not for the configurations ANYFORM and NOFORM.

cases will provide) useful information for other combinations of source and target forms.

The first type of configuration is ANYFORM: each of the available source forms in the subset of the English paradigm (*lift*, *lifts*, *lifted*) contains enough information for a correct generation of the target form *lifting*. The second configuration is SINGLEFORM: there is a single form that contains enough information for correct generation, but it has to be carefully selected. Inflected forms of the German verb *treffen* 'to meet' have different stem vowels (see Table 1). In single-source reinflection, producing a target form with one stem vowel (*a* in *trafe* in the figure) from a source form with another stem vowel (e.g., *e* in *treffe*) is difficult.[3]

In contrast, the learning problem for the SINGLEFORM configuration is much easier in multi-source MRI. The multi-source model does not have to learn the possible vowel changes of this irregular verb; instead, it just needs to pick the correct vowel change from the alternatives offered in the input. This is a relatively easy task since the theme vowel is identical. So we only need to learn one general fact about German morphology (which suffix to add) and will then be able to produce the correct form with high accuracy. This type of regularity is typical of complex morphology: there are groups of forms in a paradigm that are similar and it is highly predictable which of these groups a particular target form for a new word will be a member of. As long as one representative of each group is part of the multi-source input, we can select it to generate the correct form.

In the MULTISOURCE configuration, we are able to use information from multiple forms if no single form is sufficient by itself. For example, to generate *ponga*, 3rdSgSubPres of *poner* 'to put' in Spanish, we need to know what the stem is (*ponga*, not *pona*) and which conjugation class (*-ir*, *-er* or *-ar*) it is part of (*ponga*, not *pongue*). The single-source input *pongo*, 1stSgIndPres, does not reveal the conjugation class: it is compatible with both *ponga* and *pongue*. The single-source input *poner*, Inf, does not reveal the stem for the subjunctive: it is compatible with both *ponga* and *pona*—we need both source forms to generate the correct form *ponga*.

Again, such configurations are frequent cross-linguistically, either in this "discrete" variant or in more fuzzy variants where taking several forms together increases our chances of producing the correct target form. Finally, we call configurations NOFORM if the target form is completely irregular and not related to any of the source forms. The suppletive form *went* is our example for this case.

2.2 Principle Parts

The intuition behind the MRI task draws inspiration from the theoretical linguistic notion of **principle parts** (Finkel and Stump, 2007; Stump and Finkel, 2013). The notion is that a paradigm has a subset that allows for maximum predictability. In terms of language pedagogy, the principle parts would be a minimial set of forms a student has to learn in order to be able to generate any form

[3] It is not impossible to learn, but *treffen* is an irregular verb, so we cannot easily leverage the morphology we have learned about other verbs.

in the paradigm. For instance for the partial German paradigm in Table 1, the forms *treffen*, *trifft*, *trafen*, and *träfen* could form *one* potential set of principle parts.

From a computational learning point of view, maximizing predictability is always a boon—we want to make it as easy as possible for the system to learn the morphological regularities and subregularities of the language. Giving the system the principle parts as input is one way to achieve this.

3 Model Description

Our model is a multi-source extension of MED, Kann and Schütze (2016b)'s encoder-decoder network for MRI. In MED, a single bidirectional recurrent neural network (RNN) encodes the input. In contrast, we use multiple encoders to be able to handle multiple source form-tag pairs. In MED, a decoder RNN produces the output from the hidden representation. We do not change this part of the architecture, so there is still a single decoder.[4]

3.1 Input and Output Format

For k source forms, our model takes k different inputs of parallel structure. Each of the $1 \leq i \leq k$ inputs consists of the target tag t_{trg} and the source form f_i and its corresponding source tag t_i. The output is the target form. Each source form is represented as a sequence of characters; each character is represented as an embedding. Each tag—both the target tag and the source tags—is represented as a sequence of subtags; each subtag is represented as an embedding.

More formally, we define the alphabet Σ_{lang} as the set of characters in the language and Σ_{subtag} as the set of subtags that occur as part of the set of morphological tags $\mathcal{T}$ of the language, e.g., if 1stSgPres $\in \mathcal{T}$, then 1st, Sg and Pres $\in \Sigma_{\text{subtag}}$. Each of the k inputs to our system is of the following format: $S_{\text{start}}\Sigma^+_{\text{subtag}}\Sigma^+_{\text{lang}}\Sigma^+_{\text{subtag}}S_{\text{end}}$ where the first subtag sequence is the source tag t_i and the second subtag sequence is the target tag. The output format is: $S_{\text{start}}\Sigma^+_{\text{lang}}S_{\text{end}}$, where the symbols S_{start} and S_{end} are predefined start and end symbols.

3.2 Multi-Source Encoder-Decoder

The encoder-decoder is based on the machine translation model of Bahdanau et al. (2015) and all specifics of our model are identical to the original presentation unless stated otherwise.[5] Whereas Bahdanau et al. (2015)'s model has only one encoder, our model consists of $k \geq 1$ encoders and processes k sources simultaneously. The k sources have the form $X_m = (t_{\text{trg}}, f^m_{\text{src}}, t^m_{\text{src}})$, represented as $S_{\text{start}}\Sigma^+_{\text{subtag}}\Sigma^+_{\text{lang}}\Sigma^+_{\text{subtag}}S_{\text{end}}$ as described above. Characters and subtags are embedded.

The input to encoder m is X_m. Each encoder consists of a bidirectional RNN that computes a hidden state h_{mi} for each position, the concatenation of forward and backward hidden states. Decoding proceeds as follows:

$$p(y \mid X_1, \ldots, X_k) = \prod_{t=1}^{|Y|} p(y_t \mid \{y_1, ..., y_{t-1}\}, c_t) = \prod_{t=1}^{|Y|} g(y_{t-1}, s_t, c_t), \quad (2)$$

where $y = (y_1, ..., y_{|Y|})$ is the output sequence (a sequence of $|Y|$ characters), g is a nonlinear function, s_t is the hidden state of the decoder and c_t is the sum of the encoder states h_{mi}, weighted by attention weights $\alpha_{mi}(s_{t-1})$ that depend on the decoder state:

$$c_t = \sum_{m=1}^{k} \sum_{i=1}^{|X_m|} \alpha_{mi}(s_{t-1}) h_{mi}. \quad (3)$$

A visual depiction of this model may be found in Figure 2. A more complex hierarchical attention structure would be an alternative, but this simple model in which all hidden states contribute on the same level in a single attention layer (i.e., $\sum_{m=1}^{k} \sum_{i=1}^{|X_m|} \alpha_{mi} = 1$) works well as our experiments show. The k encoders share their weights.

4 Multi-Source Reinflection Experiment

We evaluate the performance of our model in an experiment based on Task 2 of the SIGMORPHON Shared Task on Morphological Reinflection (Cotterell et al., 2016). This is a single-source MRI task as outlined in Section 1.

4.1 Experimental Settings

Datasets. Our datasets are based on the data from the SIGMORPHON 2016 Shared Task on Morphological Reinflection (Cotterell et al.,

[4] The edit tree (Chrupała, 2008; Müller et al., 2015) augmentation discussed in Kann and Schütze (2016b) was not employed here.

[5] We modify the implementation of the model freely available at `https://github.com/mila-udem`.

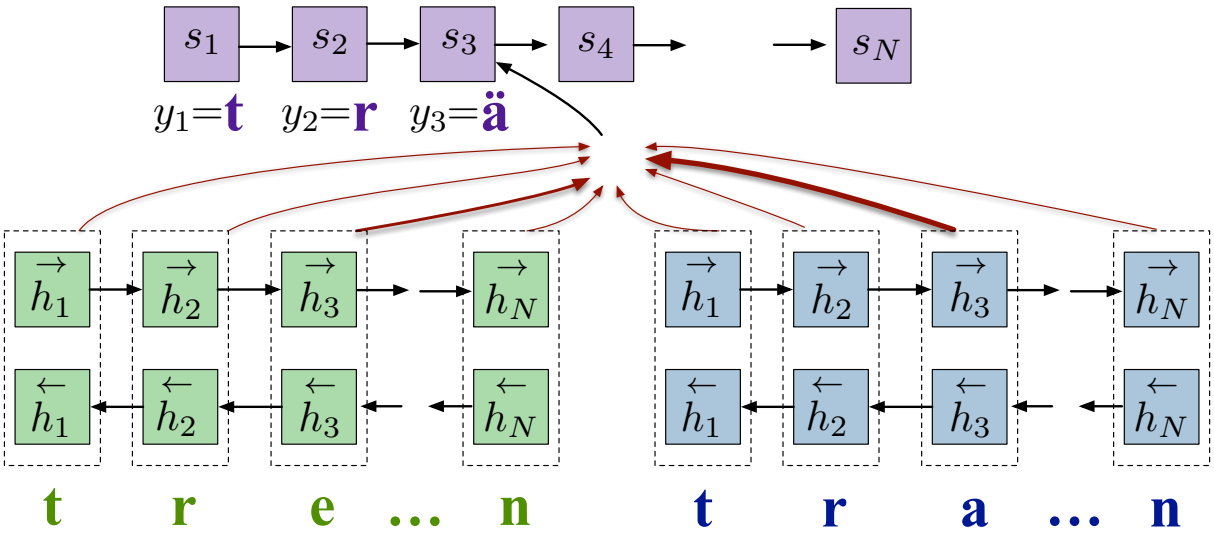


Figure 2: Visual depiction of our multi-source encoder-decoder RNN. We sketch a two encoder model, where the left encoder reads in the present form treffen and the right encoder reads in the past tense form trafen. They work together to predict the subjunctive form träfen. The shadowed red arcs indicate the strength of the attention weights—we see the network is focusing more on **a** because it helps the decoder better predict **ä** than **e**. We omit the source and target tags as input for conciseness.

2016). Our experiments cover eight languages: Arabic, Finnish, Georgian, German, Hungarian, Russian, Spanish and Turkish. The languages were chosen to represent different types of morphology. Finnish, German, Hungarian, Russian, Turkish and Spanish are all suffixing. In addition to being suffixing, three of these languages employ vocalic (German, Spanish) and consonantal (Russian) stem changes for many inflections. The members of the remaining sub-group are agglutinative. Georgian makes use of prefixation as well as suffixation. Arabic morphology contains both concatenative and templatic elements. We build multi-source versions of the dataset for Task 2 of the SIGMORPHON shared task in the following way. We use data from the UNIMORPH project,[6] containing complete paradigms for all languages of the shared task. The shared task data was sampled from the same set of paradigms; our new dataset is a superset of the SIGMORPHON data.

We create our new dataset by uniformly sampling three additional word forms from the paradigm of each source form in the original data. In combination with the source and target forms of the original dataset, this means that our dataset is a set of 5-tuples consisting each of four source forms and one target form.[7] Ideally, we would like to keep the experimental variable k, the number of sources we use in multi-source MRI, constant for a particular experiment or vary it systematically across other experimental conditions. Table 2 gives an overview of the number of different source forms per language in our dataset. Our dataset is available for download at `http://cistern.cis.lmu.de`.

	1	2	3	≥ 4
ar	0	0	0	12,800
fi	0	0	0	12,800
ka	1015	84	2	11,699
de	0	0	0	12,800
hu	0	0	0	19,200
ru	0	0	5	12,794
es	1575	25	877	10,323
tu	0	0	0	12,800

Table 2: Number of target forms in the training set for which 1, 2, 3 or ≥ 4 source forms (in the training set) are available for prediction. The tables for the development and test splits show the same pattern and are omitted.

Hyperparameters. We use embeddings of size 300. Our encoder and decoder GRUs have 100 hidden units each. Following Le et al. (2015), we initialize all encoder and decoder weights as well as the embeddings with an identity matrix. All biases are initialized with zero. We use stochas-

[6] `http://unimorph.org`

[7] One thing to note is that the original shared task data was sampled depending on word frequency in unlabeled corpora. We do not impose a similar condition, so the frequency distributions of our data and the shared task data are different. Also, we excluded Maltese and Navajo due to a lack of data to create the additional multi-source datasets.

	source form(s) used					
	1	2	3	4	1–2	1–4
ar	.871	.813	.796	.830	.905	**.944**
fi	.956	.929	.941	.934	.965	**.978**
ka	.967	.943	.942	.934	.969	**.979**
de	.954	.922	.931	.912	.959	**.980**
hu	**.992**	.962	.963	.963	.988	.989
ru	.876	.795	.824	.817	.888	**.911**
es	.975	.961	.963	.968	.977	**.984**
tu	.967	.928	.947	.944	.970	**.983**

Table 3: Accuracy on MRI for single-source (1, 2, 3, 4) and multi-source (1–2, 1–4) models. Best result in bold.

tic gradient descent, Adadelta (Zeiler, 2012) and a minibatch size of 20 for training. Training is done for a maximum number of 90 epochs. If no improvement occurs for 20 epochs, we stop training early. The final model we run on test is the model that performs best on the development data.

Baselines. For the single-source case, we apply MED, the top-scoring system in the SIGMORPHON 2016 Shared Task on Morphological Reinflection (Cotterell et al., 2016; Kann and Schütze, 2016b). At the time of writing, MED constitutes the state of the art on the dataset. For Arabic, German and Turkish, we run an additional set of experiments to test two additional architectural configurations of multi-source encoder-decoders: (i) In addition to the default configuration in which all encoders share parameters, we also test the option of each encoder learning its own set of parameters (shared par's: yes vs. no in Table 4). (ii) Another way of realizing a multi-source system is to concatenate all sources and give this to an encoder-decoder with a single encoder as one input (encoders: $k = 1$ vs. $k > 1$ in Table 4).

Evaluation Metric. We evaluate on 1-best accuracy (exact match) against the gold form. We deviate from the shared task, which also evaluates under mean reciprocal rank and edit distance. We omit the later two since all these metrics were highly correlated (Cotterell et al., 2016).

4.2 Results

Table 3 shows the results of the MRI experiment on test data. We compare using a single source, the first two sources and all four sources. The first source (in column "1") is the original source from the SIGMORPHON shared task. Recall that we used uniform sampling to identify additional forms whereas the sampling procedure of the shared task took into account frequency. We suspect that this is the reason for the worse performance of the new sources compared to the original source; e.g., in German there are rarely used subjunctive forms like *befähle* that are unlikely to help generate related forms that are more frequent.

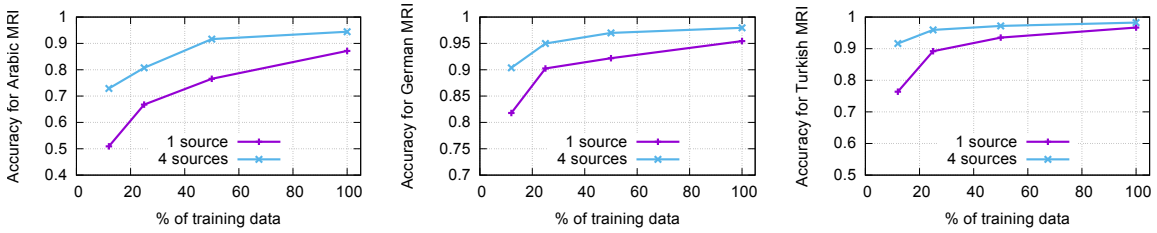
encoders:		$k = 1$	$k = 4$	
par's shared:			yes	no
	ar	**.944**	**.944**	.920
	de	**.980**	**.980**	.975
	tu	**.985**	.983	.969

Table 4: Accuracy of different architectures for the dataset with 4 source forms being available for prediction. The best result for each row is in bold.

The main result of the experiment is that multi-source MRI performs better than single-source MRI for all languages except for Hungarian and that, clearly, the more sources the better: using four sources is always better than using two sources. This result confirms our hypothesis, illustrated in Figure 1, that for most languages, different source forms provide complementary information when generating a target form and thus performance of the multi-source model is better than of the single-source model. Table 3 demonstrates that the two configurations we identified as promising for multi-source MRI, SINGLEFORM and MULTIFORM, occur frequently enough to boost the performance for seven of the eight languages, with the largest gains observed for Arabic (7.3%) and Russian (3.5%) and the smallest for Spanish (0.9%) and Georgian (1.3%) (comparing using source form 1 with using source forms 1–4).

Hungarian is the only language for which performance decreases, by a small amount (0.3%). We attribute this to overfitting: the multi-source model has a larger number of parameters, so it is more prone to overfitting. We would expect the performance to be the same in a comparison of two models that have the same size.

Error Analysis. We compare errors of single-source and multi-source models for German on development data. Most mistakes of the multi-source model are stem-related: *versterbst* for *verstirbst*, *erwerben* for *erwürben*, *Apfelsinenbaume* for *Apfelsinenbäume*, *lungenkränkes* for *lungenkrankes* and *übernehmte* for *übernähme*. In most of these cases, the stem of the lemma was



(a) Arabic (b) German (c) Turkish

Figure 3: Learning curves for single-source and multi-source models for Arabic, German and Turkish. We observe that the multi-source model generalizes faster than the single soure case—this is to be expected since the multi-source model often faces an easier transduction problem.

used, which is correct for some forms, but not for the form that had to be generated. In one case, the multi-source model did not use the correct inflection rule: *braucht* for *gebraucht*—the inflectional rule that the past participle is formed by *ge-* was not applied.

Errors of the single-source model that were "corrected" by the multi-source model include *empfahlt* for *empfiehl*, *Throne* for *Thron* and *befielen* for *befallen*. These are all SINGLEFORM cases: the multi-source model will generate the correct form if it succeeds in selecting the most predictive source form. The single-source model is at a disadvantage if this most predictive source form is not part of its input.

4.3 Comparison of Different Architectures

Table 4 compares different architectural configurations. All experiments use 4 sources. We see that sharing parameters is superior as expected. Using a single encoder on 4 sources performs as well as 4 encoders (and very slightly better on Turkish). Apparently, it has no difficulty learning to understand an unstructured (or rather lightly structured) concatenation of form-tag pairs; on the other hand, this parsing task, i.e., learning to parse the sequence of form-tag pairs, is easy, so this is not a surprising result.

4.4 Learning Curves

Figure 3 shows learning curves for Arabic, German and Turkish. We iteratively halve the training set and train models for each subset. In this analysis, we train all models for 90 epochs, but use the numbers from the main experiment for the full training set. For the single-source model, we use the SIGMORPHON source. The figure shows that the single-source model needs more individual paradigms in the training data to achieve the same performance as the multi-source model. The largest difference between single-source and multi-source is $> 20\%$ for Arabic when only 1/8 of the training set is used. This suggests that multi-source MRI is an attractive option for low-resource languages since it exploits available data better than single-source.

4.5 Attention Visualization

Figure 4 shows for one example, the generation of the German form *wögen*, 3rdPlSubPst, the attention weights of the multi-source model at each time step of the decoder, i.e., for each character as it is being produced by the decoder. For characters that simply need to be copied, the main attention lies on the corresponding characters of the input sources. For example, the character *g* is produced when attention is on the characters *g* in *wögest*, *wöge* and *wogen*. This aspect of the multi-source model is not different from the single-source model, offering no advantage.

However, even for *g*, the source form that is least relevant for generating *wögen* receives almost no weight: *wägst* is an indicative singular form that does not provide helpful information for generating a plural form in the subjunctive; the model seems to have learned that this is the case. In contrast, *wogen* does receive some weight; this makes sense as it is a past indicative form and the past subjunctive is systematically related to the past indicative for many German verbs. These observations suggest that the network has learned to correctly predict (at least in this case) which forms provide potentially useful information. For the last two time steps (i.e., characters to be generated), attention is mainly focused on the tags. Again, this indicates that the model has learned the regularity in generating this part of the word form: the suffix, consisting of *en*, is predictable from the tag.

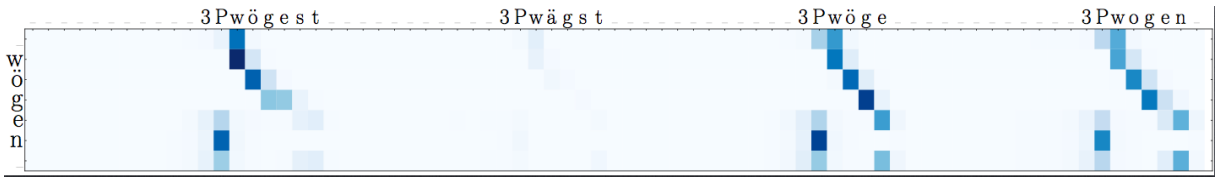


Figure 4: Attention heatmap for the multi-source model. The example is for the German verb *wiegen* 'to weigh'. The model learns to focus most of its attention on forms that share the irregular subjunctive stem *wög* in addition to the target subtags *3* and *P* that encode that the target form is 3rd person plural. We omit the tags from the diagram to which the model hardly attends.

5 Related Work

Recently, variants of the RNN encoder-decoder have seen widespread adoption in many areas of NLP due to their strong performance. Encoder-decoders with and without attention have been applied to tasks such as machine translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015), parsing (Vinyals et al., 2015) and automatic speech recognition (Graves and Schmidhuber, 2005; Graves et al., 2013).

The first work on multi-source models was presented for machine translation. Zoph and Knight (2016) made simultaneous use of source sentences in multiple languages in order to find the best match possible in the target language. Unlike our model, they apply transformations to the hidden states of the encoders that are input to the decoder. Firat et al. (2016)'s neural architecture for MT translates from any of N source languages to any of M target languages, using language specific encoders and decoders, but sharing one single attention-mechanism. In contrast to our work, they obtain a single output for each input.

Much ink has been spilled on morphological reinflection over recent years. Dreyer et al. (2008) develop a high-performing weighted finite-state transducer for the task, which was later hybridized with an LSTM (Rastogi et al., 2016). Durrett and DeNero (2013) apply a semi-CRF to heuristically extracted rules to generate inflected forms from lemmata using data scraped from Wiktionary. Improved systems for the Wiktionary data were subsequently developed by Hulden et al. (2014), who used a semi-supervised approach, and Faruqui et al. (2016), who used a character-level LSTM. All of the above work has focused on the single input case. Two important exceptions, however, have considered the multi-input case. Both Dreyer and Eisner (2009) and Cotterell et al. (2015b) define a string-valued graphical model over the paradigm and apply the missing values.

The SIGMORPHON 2016 Shared Task on Morphological Reinflection (Cotterell et al., 2016), based on the UNIMORPH (Sylak-Glassman et al., 2015) data, resulted in the development of numerous methods. RNN encoder-decoder models (Aharoni et al., 2016; Kann and Schütze, 2016a; Östling, 2016) obtained the strongest performance and are the current state of the art on the task. The best-performing model made use of an attention mechanism (Kann and Schütze, 2016a), first popularized in machine translation (Bahdanau et al., 2015). We generalize this architecture to the multi-source case in this paper for the reinflection task.

Besides generation, computational work on morphology has also focused on analysis. In this area, a common task—morphological segmentation—is to break up a word into its sequence of constituent morphs. The unsupervised MORFESSOR model (Creutz and Lagus, 2002) has achieved widespread adoption. Bayesian methods have also proven themselves successful in unsupervised morphological segmentation (Johnson et al., 2006; Goldwater et al., 2009). When labeled training data for segmentation is available, supervised methods significantly outperform the unsupervised techniques (Ruokolainen et al., 2013; Cotterell et al., 2015a).

As we pointed out in Section 2, morphologically annotated corpora provide an ideal source of data for the multi-source MRI task: they are annotated on the token level with inflectional features and often contain several different inflected forms of a lemma. Eskander et al. (2013) develop an algorithm for automatic learning of inflectional classes and associated lemmas from morphologically annotated corpora, an approach that could be usefully combined with our multi-source MRI framework.

6 Conclusion

Generation of unknown inflections in morphologically rich languages is an important task that re-

mains unsolved. We provide a new angle on the problem by considering systems that are allowed to have multiple inflected forms as input. To this end, we define the task of multi-source morphological reinflection as a generalization of single-source MRI (Cotterell et al., 2016) and present a model that solves the task. We extend an attention-based RNN encoder-decoder architecture from the single-source case to the multi-source case. Our new model consists of multiple encoders, each receiving one of the inputs. Our model improves over the state of the art for seven out of eight languages, demonstrating the promise of multi-source MRI. Additionally, we publically release our implementation.[8]

7 Future Work

The new dataset for multi-source morphological reinflection that we release is a superset of the dataset of the SIGMORPHON 2016 Shared Task on Morphological Reinflection to facilitate research on morphological generation. One focus of future work should be the construction of more complex datasets, e.g., datasets that have better coverage of irregular words and datasets in which there is no overlap in lemmata between training and test sets. Further, for difficult inflections, it might be interesting to find an effective way to include unsupervised data into the setup. For example, we could define one of our k inputs to be a form mined from a corpus that is not guaranteed to have been correctly tagged morphologically, but likely to be helpful.

We show in this paper that multi-source MRI outperforms single-source MRI. This is an important contribution because—as we discussed in Section 2.1—multi-source MRI is only promising for paradigms with specific properties, which we referred to as SINGLEFORM and MULTIFORM configurations. Whether such configurations occur and whether these configurations have a strong effect on MRI performance was an open empirical question. Indeed, we found that for one of the languages we investigated, for Hungarian, single-source MRI works at least as well as multi-source MRI—presumably because its paradigms almost exclusively contain SINGLEFORM configurations. Thus, single-source MRI is probably preferable for Hungarain since single-source is simpler than multi-source.

[8] http://cistern.cis.lmu.de

There is another important question that we have not answered in this paper: in an experimental setting in which the amount of training information available is exactly the same for single-source and multi-source, does multi-source still outperform single-source and by how much? For example, the numbers we compare in Table 3 are matched with respect to the number of target forms, but not with respect to the number of source forms: multi-source has more source forms available for training than single-source. We leave investigation of this important issue for future work.

Acknowledgments

We gratefully acknowledge the financial support of Siemens and of DFG (SCHUE 2246/10-1) for this research. The second author was supported by a DAAD Long-Term Research Grant and an NDSEG fellowship.

References

Roee Aharoni, Yoav Goldberg, and Yonatan Belinkov. 2016. Improving sequence to sequence learning for morphological inflection generation: The BIU-MIT systems for the SIGMORPHON 2016 shared task for morphological reinflection. In *2016 Meeting of SIGMORPHON*.

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

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

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

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 conference on machine translation. In *WMT*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Grzegorz Chrupała. 2008. *Towards a machine-learning architecture for lexical functional grammar parsing*. Ph.D. thesis, Dublin City University.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015a. Labeled morphological segmentation with semi-markov models. In *CoNLL*.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2015b. Modeling word forms using latent underlying morphs and phonology. In *TACL*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *2016 Meeting of SIGMORPHON*.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *ACL Workshop on Morphological and Phonological Learning*.

Markus Dreyer and Jason Eisner. 2009. Graphical models over multiple strings. In *EMNLP*.

Markus Dreyer, Jason R Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *EMNLP*.

Markus Dreyer. 2011. *A non-parametric model for the discovery of inflectional paradigms from plain text using graphical models over strings*. Ph.D. thesis, Johns Hopkins University, Baltimore, MD.

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

Ramy Eskander, Nizar Habash, and Owen Rambow. 2013. Automatic extraction of morphological lexicons from morphologically annotated corpora. In *EMNLP*.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *NAACL*.

Raphael Finkel and Gregory Stump. 2007. Principal parts and morphological typology. *Morphology*, 17(1):39–75.

Orhan Firat, KyungHyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *CoRR*.

Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. 2009. A bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610.

Alan Graves, Abdel-Rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *ICASSP*.

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

Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *NIPS*.

Katharina Kann and Hinrich Schütze. 2016a. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *2016 Meeting of SIGMORPHON*.

Katharina Kann and Hinrich Schütze. 2016b. Single-model encoder-decoder with explicit morphological representation for reinflection. In *ACL*.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Ryan T McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, and Oscar Täckström. 2013. Universal dependency annotation for multilingual parsing. In *ACL*.

Thomas Müller, Ryan Cotterell, and Alexander Fraser. 2015. Joint lemmatization and morphological tagging with LEMMING. In *EMNLP*.

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

Robert Östling. 2016. Morphological reinflection with convolutional neural networks. In *2016 Meeting of SIGMORPHON*.

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

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *CoNLL*.

Gregory Stump and Raphael A Finkel. 2013. *Morphological typology: From word to paradigm*, volume 138. Cambridge University Press.

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

John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015. A language-independent feature schema for inflectional morphology. In *ACL-IJCNLP*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *NIPS*.

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

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *arXiv preprint arXiv:1601.00710*.

Chapter 7

The LMU System for the CoNLL-SIGMORPHON 2017 Shared Task on Universal Morphological Reinflection

The LMU System for the CoNLL-SIGMORPHON 2017 Shared Task on Universal Morphological Reinflection

Katharina Kann and **Hinrich Schütze**
CIS
LMU Munich, Germany
kann@cis.lmu.de

Abstract

We present the LMU system for the CoNLL-SIGMORPHON 2017 shared task on universal morphological reinflection, which consists of several subtasks, all concerned with producing an inflected form of a paradigm in different settings. Our solution is based on a neural sequence-to-sequence model, extended by preprocessing and data augmentation methods. Additionally, we develop a new algorithm for selecting the most suitable source form in the case of multi-source input, outperforming the baseline by 5.7% on average over all languages and settings. Finally, we propose a fine-tuning approach for the multi-source setting, and combine this with the source form detection, increasing accuracy by a further 4.6% on average.

1 Introduction

Many of the world's languages have a rich morphology, i.e., make use of surface variations of lemmata in order to express certain properties, like the tense or mood of a verb. This makes a variety of natural language processing tasks more challenging, as it increases the number of words in a language drastically; a problem morphological analysis and generation help to mitigate. However, a big issue when developing methods for morphological processing is that for many morphologically rich languages, there are only few or no relevant training data available, making it impossible to train state-of-the-art machine learning models (e.g., (Faruqui et al., 2016; Kann and Schütze, 2016b; Aharoni et al., 2016; Zhou and Neubig, 2017)). This is the motivation for the CoNLL-SIGMORPHON-2017 shared task on universal morphological reinflection (Cotterell et al., 2017a), which animates the development of systems for as many as 52 different languages in 6 different low-resource settings for morphological reinflection: to generate an inflected form, given a target morphological tag and either the lemma (task 1) or a partial paradigm (task 2). An example is

(use, V;3;SG;PRS) $\mapsto$ uses

In this paper, we describe the LMU system for the shared task. Since it depends on the language and the amount of resources available for training which method performs best, our approach consists of a modular system. For most medium- and high-resource, as well as some low-resource settings, we make use of the state-of-the-art encoder-decoder (Cho et al., 2014a; Sutskever et al., 2014; Bahdanau et al., 2015) network MED (Kann and Schütze, 2016b), while extending the training data in several ways. Whenever the given data are not sufficient, we make use of the baseline system, which can be trained on fewer instances.

While we submit solutions for every language and setting, our main focus is on task 2 of the shared task and the main contributions of this paper correspondingly address a multi-source input setting: (i) We develop CIS ("choice of important sources"), a novel algorithm for selecting the most appropriate source form for a target tag from a partially given paradigm, which is based on edit trees (Chrupała, 2008). (ii) We propose to cast the task of multi-source morphological reinflection as a domain adaptation problem. By fine-tuning on forms from a partial paradigm, we improve the performance of a neural sequence-to-sequence model for most shared task languages.

Our final methods, averaged over languages, outperform the official baseline by 7.0%, 18.5%, and 16.5% for task 1 and 8.7%, 10.1%, and

10.3% for task 2 for the low-, medium-, and high-resource settings, respectively.

Furthermore, our submitted sytem—a combination of our methods with the baseline system—surpasses the baseline's accuracy on test for both tasks as well as all languages and settings. Differences in performance are between 8.69% (task 1 low) and 17.94% (task 1 medium).

2 Morphological Reinflection

The paradigm of a lemma w_l is a set of tuples of inflected forms f_k and tags t_k describing the properties of the inflected word, which we formally denote as:

$$\pi(w_l) = \left\{ \left(f_k[w_l], t_k \right) \right\}_{t_k \in T(w_l)} \quad (1)$$

with $T(w_l)$ being the set of possible tags for w_l.

An example is the following paradigm of the Spanish lemma *soñar*:

$$\pi(\text{soñar}) = \left\{ \left(\textit{sueño}, \text{1SgPresInd} \right), \ldots, \left(\textit{soñaran}, \text{3PlPastSbj} \right) \right\}$$

The shared task has two subtasks: **task 1** consists of predicting a certain form $f_i[w_l]$, given the lemma w_l and the target tag t_i. For **task 2**, one or more source forms are given for each lemma (multi-source input). Thus, additional information about the way a lemma is inflected is known and can be leveraged.

3 Preprocessing Methods

We apply the following preprocessing methods.

String preprocessing. We determine for each language if it is predominantly prefixing or suffixing, using the same algorithm as the shared task baseline system (Cotterell et al., 2017a). For prefixing languages, we invert all words. An example for the prefixing language Navajo is:

chidí → ídihc

New character handling. The source and target vocabularies for the languages are constructed using the respective training and development sets. Therefore, out-of-vocabulary symbols can appear in the test sets, resulting in symbols the model has no information about. In order to address this, we substitute such characters by a special `NEW` symbol and train the model on it by including it in the additional training samples we create, cf. §4. In the output, `NEW` is substituted back by the new characters in the input in order of appearance. An example from the German development data is:

Phloëm → Phlo`NEW`m

Tag extension. Explicit information is usually handled better by machine learning methods than implicit information. Therefore, we search for optional subtag slots, in contrast to those that are always occupied by some value, e.g., an optional *negation* subtag, in contrast to the part-of-speech subtag which, for most languages, is always either *Verb*, *Noun* or *Adjective*, but never empty. For all optional subtags, we artificially introduce a negated form.

4 Training Data Augmentation Methods

Additional source-target form pairs. We collect all forms belonging to the same lemma. We then add additional samples by constructing source-target combinations for other sources than the lemma, using the members of each paradigm. For the two samples $lemma_i \rightarrow word_1$ and $lemma_i \rightarrow word_2$ we can introduce the new samples $word_1 \rightarrow word_2$ and $word_2 \rightarrow word_1$.[1]

Autoencoding samples. We further create samples for a sequence autoencoding task, i.e., we add mappings of words to themselves, with a special copy tag **`A`**. No morphological tags are given. This is a way to multi-task train on autoencoding the input string and reinflection, as we maximize the joint log-likelihood

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(w_l, t_s, t_t) \in \mathcal{T}} \log p_{\boldsymbol{\theta}} \left(f_t(w_l) \mid e(f_s(w_l), t_t) \right) + \sum_{w \in \mathcal{W}} \log p_{\boldsymbol{\theta}}(w \mid e(w)) \quad (2)$$

for the training data $\mathcal{T}$, source and target tags t_s and t_t, a lemma w_l and an encoding function e depending on $\boldsymbol{\theta}$, as well as a set of strings $\mathcal{W}$. We apply two variants: autoencoding the lemmata and forms from the original training set, or using *random* strings for this. Random strings are produced in the following way. We first construct all possible bigrams $\mathcal{B}$ from the vocabulary of the language. We then combine those with a random sequence of characters r of a random length between

[1] The respective source and target tags are part of the input, but omitted here for clarity.

1 and 4 in the following way: $b_1 + b_2 + r + b_3 + b_4$ for $b_i \in \mathcal{B}$. Constructing random strings like this has the positive side-effect that we can add a `NEW` to the vocabulary.

Rule-based data generation. We imitate a rule-based system by, given a source form and a target form, defining the prefix (resp. suffix) of a word as the word minus the longest common suffix (resp. prefix). We then create an additional training example by generating a random string s and prepending (resp. appending) source and target prefixes (resp. suffixes) to s. For example, in German, we can find the following rule for the 2nd person singular form:

$$\text{*en} \rightarrow \text{*st}$$

From this we can create additional training instances like the following.

$$\text{(jfgdgfen, V;2;SG;PRS)} \mapsto \text{jfgdgfst}$$
$$\text{(Ahggen, V;2;SG;PRS)} \mapsto \text{Ahggst}$$

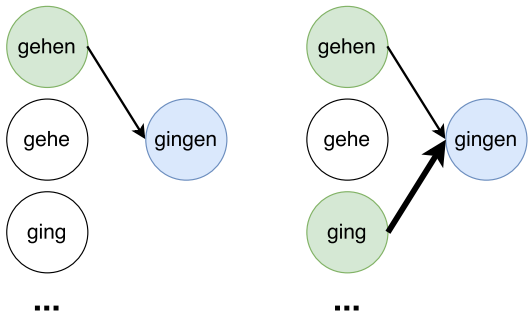
We apply this procedure to all pairs of a source and a target tag that appear less than t times in train for a certain threshold t.

5 System Architecture

We apply the encoder-decoder network MED (Kann and Schütze, 2016a), due to its success in last year's edition of the shared task (Cotterell et al., 2016). While we extend it by new training data augmentation methods and, for task 2, the additional algorithms described below, we do not make changes to the model's architecture. We will shortly describe MED and the shared task baseline system in this section.

5.1 MED

Encoder. The format of the input of the encoder is the same as in (Kann and Schütze, 2016a), but with a small modification to be able to handle unlabeled data: Given the set of morphological subtags M that each target tag is composed of (e.g., the tag *1SgPresInd* contains the subtags *1*, *Sg*, *Pres* and *Ind*), and the alphabet Σ of the language of application, our input is of the form $(\mathbf{A} \mid \text{M}^*)\ \Sigma^*$, i.e., it consists of *either* a sequence of subtags *or* the symbol **A** signaling that the input is not annotated and should be autoencoded, and (in both cases) the character sequence of the input word. All parts of the input are represented by embeddings.



(a) The lemma is the only accepted source form.

(b) Additional source forms with a higher priority than the lemma have been determined.

Figure 1: Comparison of the traditional view (left) and the result of CIS (right). Possible source forms in green, the target form in blue. Thickness of the arrows represents priorities of source forms. Most forms of the paradigm have been omitted because of space limitations.

We encode the input $x = x_1, x_2, \ldots, x_{T_x}$ using a bidirectional gated recurrent neural network (GRU) (Cho et al., 2014b). We then concatenate the forward and backward hidden states to obtain the input h_i for the decoder.

Decoder. The decoder is a uni-directional attention-based GRU, defining a probability distribution over strings in Σ^*:

$$p(y \mid x) = \prod_{t=1}^{T_y} p(y_t \mid y_1, \ldots, y_{t-1}, s_t, c_t),$$

with s_t being the decoder hidden state for time t and c_t being a context vector, calculated using the encoder hidden states together with attention weights. A detailed description of the encoder-decoder model can be found in (Bahdanau et al., 2015).

5.2 Baseline System

The shared task baseline system (**BL**) is well-suited for low-resource settings. It first aligns each input and output string, and than extracts possible prefix or suffix substitution rules from the training data. At test time, it applies the most suitable one in the following way: Every input is searched for the longest contained prefix or suffix and the rule belonging to the affix and given target tag is applied to obtain the output. Whether prefixes or suffixes are used depends on the language and is determined using the training set.

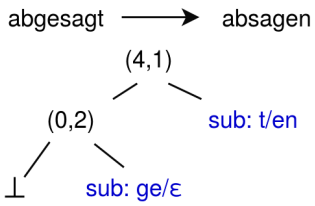


Figure 2: Edit tree for the transformation from *abgesagt* "canceled" to *absagen* "to cancel". Each node contains the length of the parts before and after the respective LCS, e.g., the leftmost node contains the length of the parts before and after the LCS of *abge* and *ab*. The prefix *sub* indicates that the node is a substitution operation.

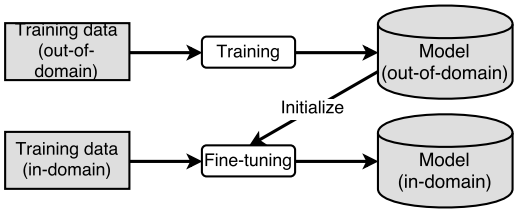


Figure 3: Overview of a fine-tuning setup. In our case, "in-domain" refers to the partial paradigm to be completed; "out-of-domain" refers to all other paradigms.

6 Choice of Important Sources

As our **choice of important sources (CIS) algorithm** is based strongly on edit trees (Chrupała, 2008), we will introduce them first.

Edit trees. An edit tree $e(\sigma, \tau)$ is a way to specify a transformation between a source string σ and a target string τ (Chrupała, 2008). It is constructed by first determining the longest common substring (LCS) (Gusfield, 1997) of σ and τ and then modeling the prefix and suffix pairs of the LCS recursively. In the case of an empty LCS, $e(\sigma, \tau)$ corresponds to the substitution operation that replaces σ with τ. Figure 2 shows an example.

CIS. The entire task of paradigm completion is built upon the notion that the members of a paradigm are not independent. However, for many languages, some slots of a paradigm are more dependent on each other: For example, *gehen*, *gehe* and *ging* are all forms of the same German paradigm, but when aiming to produce the 3rd person plural past tense form *gingen*, the task is easier when starting from the (more similar) form *ging*. In fact, in many cases, the entire paradigm is *completely deterministic* when the right paradigm slots are known. A set of forms that determines all other inflected forms is called *principal parts*.

(Cotterell et al., 2017b) use this property of morphologically rich languages to induce topologies in order to jointly decode entire paradigms and to thus make use of all known forms. However, they suppose to be able to compute and use good estimates for the probabilities $p(f_i(w_l)|f_j(w_l))$ for source form $f_j(w_l)$ and target form $f_i(w_l)$, since they use at least 632 entire paradigms per part of speech and language for training. Using a minimum spanning tree, they approximate a solution to the maximum-a-posteriori (MAP) inference problem.

In order to be able to apply our approach to low-resource settings, we focus instead on finding the *best source form* for each target form in a language, and CIS works as follows. We calculate edit trees for each pair $(f_j(w_l), f_i(w_l))$ for each lemma w_l in the training data. We then count the number of different edit trees for each pair of source and target tag (t_j, t_i) and build an *importance list* for each tag t_i, giving higher priorities to source tags with lower counts. The intuition behind this is that the fewer different edit trees appear in the training set, the more deterministic the paradigm slot i is, given a certain source slot j.

At test time, we find the form from the given slots of the paradigm which has the highest importance score, and use it to generate the target form. Note that, as the lemma is always given, there will never be a need to use a worse source form than the lemma.

7 Fine-Tuning for Multi-Source Input

For sequence-to-sequence models for neural machine translation, it has been shown that specialized models for a certain domain are able to obtain better performances than general ones (Luong and Manning, 2015). One way to perform such a *domain adaptation* is fine-tuning: a general model, which has been trained on out-of-domain data, is further trained on (newly) available in-domain data, cf. Figure 3. This brings the conditional probability $p(y_1, ..., y_m|x_1, ..., x_n)$ for an output sequence $(y_1, ..., y_m)$ given an input sequence $(x_1, ..., x_n)$ closer to the target distribution.

Here, we propose to improve multi-source morphological reinflection by treating each paradigm as a separate domain and performing "domain adaptation" everytime a new paradigm should be completed by the model.

In particular, we have one base model (for

$n \leq 1.5$	$1.5 < n < 10$	$10 \leq n$
danish	arabic	albanian
english	bengali	armenian
norwegian-bokmal	bulgarian	basque
norwegian-nynorsk	czech	catalan
	dutch	haida
	estonian	hindi
	faroese	italian
	finnish	khaling
	french	persian
	georgian	portuguese
	german	quechua
	hebrew	sorani
	hungarian	spanish
	icelandic	turkish
	irish	urdu
	kurmanji	welsh
	latin	
	latvian	
	lithuanian	
	lower-sorbian	
	macedonian	
	navajo	
	northern-sami	
	polish	
	romanian	
	russian	
	scottish-gaelic	
	serbo-croatian	
	slovak	
	slovene	
	swedish	
	ukrainian	

Table 1: Average amount n of sources given per paradigm, for the development set.

each setting and language), trained on all available training examples. The original training data corresponds to *out-of-domain data* in a domain adaptation setting. At test time, we construct for each partial paradigm $\mathcal{P}_{known}$ all possible training examples in the way described in the paragraphs about additional source-target form pairs and autoencoding in §4. Thus, for $|\mathcal{P}_{known}| = n$, we end up with (up to) $n * (n-1) + N_a$ in-domain samples for fine-tuning where N_a is the number of autoencoding training samples. We then for each partial paradigm fine-tune the original base model on all examples constructed from $\mathcal{P}_{known}$, which match the *in-domain data* for domain adaptation. Thus, we end up with a different fine-tuned model for each partial paradigm in the test set.

Our method is expected to perform best in a setting in which many forms of each paradigm are given as input, e.g., when n is big. Table 1 indicates for which language we would therefore expect could performance.

8 Experiments

8.1 Systems

Task1. For task 1, we apply **MED***: MED in combination with all preprocessing methods mentioned in §3 and the following data augmentations. We create additional source-target form pairs where possible and create autoencoding samples, random ones as well as from the original data. Further, we create 5 additional rule-based samples for each existing sample of all source-target tag combinations that appear less than $t = 10$ times in the training set for a language.

We employ ensembles of 5 MED* models, which are trained for 90 (low and medium) or 45 (high) epochs. Ensembling is done by majority voting.

Task2. We again apply MED*. However, for task 2 we do not create rule-based samples.[2] Models for the low-resource, medium-resource and high-resource settings are trained for 45, 30 and 20 epochs, respectively. For task 2, we do not use ensembling.

At test time, we preprocess each newly incoming paradigm in the same way as the training data, except for the creation of random copy samples. We then *fine-tune* the base model for each new paradigm according to §7 for 25 additional epochs. Additionally, we choose the best source form for each required target tag and predict each inflected form for this input (**MED*+FT+CIS**).

The limited amount of data makes it impossible to obtain competitive performance using MED* for some languages and settings (especially for languages with only few given slots per paradigm), even after applying all data augmentation methods described above. Thus, we apply the baseline model for those cases, but combine it with CIS (cf. §6) to improve its performance (**BL+CIS**). We do not apply preprocessing or data augmentation methods for BL, as they would not influence its performance.

Shared task submission. The best approach depends on both the language and the setting. Thus, our final submission for each case is obtained by either BL, BL+CIS, the MED* ensemble, or MED*+FT+CIS, selected using the accuracy on the development set.

[2]Using rule-based examples for training leads to worse performance of the fine-tuned system, even though the base system turns out to be better. Thus, we do not use it.

	low			medium			high		
	BL	MED*	MED* (ENS)	BL	MED*	MED* (ENS)	BL	MED*	MED* (ENS)
albanian	**0.216**	0.102	0.129	0.661	0.849	**0.878**	0.781	0.966	**0.975**
arabic	0.215	0.237	**0.298**	0.400	0.804	**0.842**	0.477	0.930	**0.952**
armenian	0.378	0.444	**0.488**	0.766	0.897	**0.914**	0.891	0.972	**0.975**
bulgarian	0.331	0.437	**0.480**	0.750	0.814	**0.837**	0.900	0.969	**0.974**
catalan	0.552	0.560	**0.598**	0.832	0.903	**0.930**	0.942	0.981	**0.983**
czech	**0.408**	0.318	0.341	0.807	0.815	**0.856**	0.904	0.927	**0.937**
danish	0.598	0.636	**0.654**	0.781	0.830	**0.845**	0.891	0.934	**0.960**
dutch	**0.537**	0.500	0.521	0.717	0.828	**0.862**	0.868	0.968	**0.971**
english	0.762	0.831	**0.852**	0.902	0.928	**0.940**	0.950	0.964	**0.968**
faroese	0.307	0.347	**0.386**	0.587	0.595	**0.672**	0.747	0.817	**0.867**
finnish	**0.162**	0.120	0.147	0.425	0.682	**0.754**	0.785	0.939	**0.954**
french	0.630	0.579	**0.635**	0.761	0.789	**0.820**	0.836	0.889	**0.914**
georgian	0.712	0.802	**0.845**	0.900	0.925	**0.928**	0.940	0.991	**0.995**
german	0.537	0.541	**0.593**	0.715	0.772	**0.800**	0.812	0.894	**0.912**
hebrew	0.279	0.335	**0.366**	0.400	0.798	**0.831**	0.558	0.987	**0.991**
hindi	0.310	0.781	**0.782**	0.866	0.964	**0.974**	0.940	**1.000**	**1.000**
hungarian	0.172	0.300	**0.346**	0.417	0.708	**0.763**	0.711	0.856	**0.874**
icelandic	0.342	0.341	**0.364**	0.614	0.647	**0.689**	0.761	0.873	**0.913**
italian	0.449	0.392	**0.467**	0.738	0.920	**0.927**	0.799	**0.978**	0.974
latvian	**0.621**	0.483	0.536	0.851	0.834	**0.861**	0.910	0.965	**0.977**
lower-sorbian	0.343	0.451	**0.488**	0.705	0.788	**0.817**	0.860	0.966	**0.973**
macedonian	0.500	0.577	**0.664**	0.823	0.901	**0.913**	0.919	0.957	**0.964**
navajo	0.184	0.166	**0.198**	0.313	0.415	**0.460**	0.383	0.838	**0.897**
northern-sami	0.154	0.136	**0.174**	0.357	0.639	**0.711**	0.611	0.954	**0.968**
norwegian-nynorsk	0.508	0.489	**0.559**	0.633	0.671	**0.687**	0.783	0.883	**0.923**
persian	0.273	0.405	**0.457**	0.654	0.892	**0.913**	0.776	0.999	**1.000**
polish	0.419	0.366	**0.431**	0.752	0.751	**0.780**	0.894	0.909	**0.925**
portuguese	0.603	0.633	**0.684**	0.929	0.938	**0.944**	0.974	0.986	**0.993**
quechua	0.172	0.567	**0.615**	0.681	0.965	**0.977**	0.947	**1.000**	**1.000**
russian	**0.428**	0.319	0.366	0.750	0.763	**0.801**	0.820	0.909	**0.919**
scottish-gaelic	0.480	0.600	**0.620**	0.520	0.940	**0.960**	–	–	–
serbo-croatian	0.213	0.286	**0.324**	0.658	0.812	**0.844**	0.840	0.900	**0.920**
slovak	0.419	0.467	**0.495**	0.707	0.788	**0.795**	0.852	0.940	**0.960**
slovene	0.474	0.494	**0.522**	0.819	0.865	**0.883**	0.898	0.966	**0.981**
spanish	**0.586**	0.465	0.554	0.854	0.891	**0.910**	0.906	0.965	**0.974**
swedish	0.543	0.590	**0.607**	0.737	0.772	**0.796**	0.854	0.901	**0.914**
turkish	0.143	**0.280**	0.255	0.331	0.801	**0.852**	0.729	0.977	**0.982**
ukrainian	**0.729**	0.350	0.393	0.715	0.757	**0.775**	0.863	0.929	**0.934**
urdu	0.303	0.669	**0.687**	0.861	0.955	**0.962**	0.958	**0.996**	0.995
welsh	0.150	0.340	**0.460**	0.540	0.910	**0.920**	0.670	**0.990**	**0.990**
basque	0.000	0.140	**0.180**	0.020	0.860	**0.870**	0.060	**0.990**	**0.990**
bengali	0.440	0.610	**0.680**	0.750	**0.980**	**0.980**	0.840	**0.990**	**0.990**
estonian	0.226	0.242	**0.271**	0.624	0.796	**0.832**	0.762	0.985	**0.992**
haida	0.340	0.480	**0.570**	0.560	**0.920**	0.910	0.690	**0.970**	**0.970**
irish	**0.318**	0.188	0.222	0.447	0.626	**0.694**	0.543	0.891	**0.929**
khaling	0.039	0.157	**0.184**	0.184	0.879	**0.901**	0.538	0.995	**0.998**
kurmanji	**0.823**	0.818	0.620	0.884	0.904	**0.916**	0.922	0.934	**0.943**
latin	**0.160**	0.139	0.028	0.368	0.430	**0.489**	0.456	0.735	**0.795**
lithuanian	**0.235**	0.168	0.193	0.530	0.592	**0.618**	0.647	0.867	**0.906**
norwegian-bokmal	0.690	0.722	**0.743**	0.798	0.820	**0.838**	0.906	0.907	**0.925**
romanian	**0.441**	0.335	0.392	0.702	0.715	**0.764**	0.804	0.863	**0.893**
sorani	0.205	0.175	**0.232**	0.528	0.794	**0.823**	0.643	0.899	**0.910**
Average:	0.386	0.421	**0.456**	0.647	0.804	**0.832**	0.751	0.902	**0.916**

Table 2: Accuracies for task 1, for BL, MED* and MED* ensembles. Upper part: development languages; lower part: surprise languages.

8.2 MED Hyperparameters

We use the same hyperparameters for all MED models, i.e., all languages, tasks and amounts of resources. In particular, we keep them fixed to the following. Encoder and decoder RNNs each have 100 hidden units and the embeddings size is 300. For training we use ADADELTA (Zeiler, 2012) with minibatch size 20. Following Le et al. (2015), we initialize all weights in the encoder, decoder and the embeddings except for the GRU weights in the decoder to the identity matrix. Biases are initialized to zero. We use dropout with a coefficient of 0.5. As this is the model we use to produce test results for the shared task, we report

	Task 1	Task 2
Low	100	535
Medium	994	2285
High	9825	8578

Table 3: Average amount of training examples per task and resource quantity.

the best numbers obtained on the development set during training ("early stopping"). We compare the 1-best accuracy of all systems, i.e., the percentage of predictions that match the true answer exactly.

8.3 Data

The official shared task data consists of sets for 52 different languages, 2 tasks and 3 different settings with varying amount of resources.[3] An overview of the (averaged) amount of samples per task and setting is given in Table 3. Development and test sets are the same for all settings for each respective task and language. The gold labels for the test set are not published yet, so the experiments in this paper are performed on the development set.

8.4 Results

We compare our approaches to the official shared task baseline. Detailed results for task 1 and task 2 are shown in Table 2 and Table 4, respectively.

Task 1. Table 2 shows the results obtained by MED*, both for single models and ensembles. As can be seen, MED* already outperforms the baseline for the majority of languages in all settings; in average by 0.035, 0.157 and 0.151, respectively. MED*'s performance is worse for the low data quantity than for the others. This is an expected result, as neural networks are known to require a huge amount of training instances.

Ensembling increases the final accuracy for all settings, by an average of 0.035 (low), 0.028 (medium) and 0.014 (high).

Task 2. As can be seen in Table 4, combining BL and CIS outperforms BL on its own for many languages, especially in the low-resource setting. The highest improvements for the low-, medium- and high-resource setting are for Hungarian (0.362), Latin (0.440) and Latin (0.429), respectively. For some languages, e.g., Catalan, Danish or Urdu, choosing a good source form seems to not be important. For a few languages, results even get worse. We will discuss some of those cases in §9. Overall, however, we obtain 0.087 (low), 0.066 (medium) and 0.019 (high) improvement on average over all languages, which clearly shows the usefulness of CIS.

MED* on its own does not achieve competitive performance for task 2. We attribute this to the limited number of different lemmata given for training, resulting in an overfitting model, learning, e.g., to produce certain character combinations for certain tags. However, MED*+FT+CIS outperforms both BL as well as BL+CIS for many languages in the medium- and high-resource settings and even in some low-resource scenarios. Comparing the obtained accuracies with Table 1, it gets obvious that languages with a higher amount of given source forms per paradigm achieve better results after fine-tuning, many times reaching a higher accuracy than BL, even in the low-resource setting. In contrast, fine-tuning works poorly for languages with ≤ 1.5 given source forms per paradigm. In total, using MED*+FT+CIS, we obtain an average improvement of 0.068 (low), 0.101 (medium) and 0.103 (high) over the baseline.

8.5 Official Shared Task Evaluation

Our submitted system obtained average accuracies of 0.4659 (low), 0.8264 (medium) and 0.947 (high) for task 1, and 0.6776 (low), 0.8202 (medium) and 0.8852 (high) for task 2, respectively. This corresponds to place 5 of 18, 3 of 19 and 7 of 15 for the high-, medium- and low-resource settings of task 1, respectively. Remarkably, the difference to the best system for the two higher settings is less than 0.01.

Among 3 submissions for task 2, our system comes first. It beats the baseline by 17.16 (low), 15.54 (medium) and 10.84 (high).

9 Remaining Challenges

Certain parts of our system do not perform as well for some languages as we would expect. In this section we will discuss those cases in more detail.

CIS. For some languages, e.g., Danish or English, CIS does not influence the performance. This might be due to those languages not having paradigm slots that are regularly closer to certain slots than others.

One other problem for the algorithm are training instances that consist of multiple separate words, e.g., the edit trees for "ride a bike" $\mapsto$

[3] A list of all languages can be found in Tables 2 and 4.

	low				medium				high			
	BL	BL+ CIS	MED*	MED*+ FT+CIS	BL	BL+ CIS	MED*	MED*+ FT+CIS	BL	BL+ CIS	MED*	MED*+ FT+CIS
albanian	0.160	0.249	0.000	**0.619**	**0.882**	0.280	0.016	0.865	0.942	0.434	0.240	**0.960**
arabic	0.380	0.428	0.011	**0.706**	0.553	0.704	0.059	**0.907**	0.566	0.733	0.533	**0.953**
armenian	0.722	0.855	0.001	**0.933**	0.785	0.962	0.210	**0.969**	0.856	0.806	0.517	**0.983**
bulgarian	0.553	**0.592**	0.006	0.571	0.640	0.646	0.200	**0.747**	0.819	0.810	0.677	**0.911**
catalan	**0.942**	0.938	0.000	0.877	0.958	**0.970**	0.266	0.962	0.965	0.976	0.759	**0.992**
czech	0.307	**0.346**	0.008	0.312	0.610	**0.635**	0.160	0.580	**0.841**	0.839	0.429	0.806
danish	**0.567**	**0.567**	0.284	0.287	**0.753**	**0.753**	0.541	0.410	**0.827**	**0.827**	0.673	0.680
dutch	0.588	**0.666**	0.057	0.608	0.796	**0.932**	0.509	0.796	0.845	0.965	0.812	**0.969**
english	**0.784**	**0.784**	0.544	0.576	0.832	0.832	**0.852**	0.784	0.900	0.900	0.900	**0.924**
faroese	0.513	**0.592**	0.000	0.171	0.559	**0.674**	0.234	0.578	0.651	0.738	0.430	**0.761**
finnish	0.517	**0.629**	0.017	0.581	0.720	0.743	0.143	**0.899**	0.709	0.772	0.470	**0.948**
french	0.864	0.876	0.000	**0.877**	0.893	0.936	0.379	**0.951**	0.982	0.959	0.824	**0.983**
georgian	0.793	**0.853**	0.000	0.834	0.900	**0.922**	0.532	0.909	0.933	0.954	0.793	**0.966**
german	0.610	**0.647**	0.123	0.625	0.662	0.748	0.255	**0.764**	0.705	0.813	0.619	**0.874**
hebrew	0.380	0.683	0.012	**0.786**	0.417	0.701	0.217	**0.895**	0.547	0.743	0.596	**0.950**
hindi	0.698	0.719	0.000	**0.970**	0.746	0.867	0.040	**0.970**	0.961	0.563	0.719	**1.000**
hungarian	0.255	0.617	0.000	**0.627**	0.453	0.823	0.238	**0.824**	0.585	0.877	0.503	**0.949**
icelandic	0.439	**0.546**	0.000	0.333	0.531	**0.683**	0.083	0.588	0.617	**0.753**	0.380	0.751
italian	0.769	**0.843**	0.000	0.809	0.839	0.901	0.075	**0.927**	0.901	0.896	0.503	**0.976**
latvian	0.790	**0.839**	0.001	0.565	0.852	**0.926**	0.330	0.825	0.877	**0.953**	0.705	0.951
lower-sorbian	0.362	**0.532**	0.003	0.509	0.670	**0.811**	0.302	0.769	0.866	**0.878**	0.650	0.867
macedonian	0.396	**0.562**	0.001	0.367	0.832	**0.858**	0.175	0.740	0.942	**0.964**	0.749	0.876
navajo	0.306	**0.404**	0.008	0.313	0.385	0.502	0.088	**0.517**	0.408	0.593	0.282	**0.650**
northern-sami	0.314	**0.485**	0.000	0.243	0.499	**0.841**	0.028	0.758	0.562	0.905	0.201	**0.912**
Norwegian-nynorsk	0.439	**0.445**	0.127	0.122	**0.604**	**0.604**	0.452	0.341	**0.610**	0.579	0.560	0.555
persian	0.822	0.159	0.000	**0.990**	0.911	0.185	0.203	**0.997**	0.889	0.190	0.854	**1.000**
polish	0.506	**0.596**	0.002	0.327	0.694	**0.787**	0.170	0.704	0.794	**0.831**	0.619	0.820
portuguese	0.951	**0.973**	0.001	0.934	0.969	**0.987**	0.243	0.969	0.975	**0.995**	0.741	0.991
quechua	0.973	**1.000**	0.000	0.972	0.973	0.973	0.234	**0.996**	0.972	**0.999**	0.796	**0.999**
russian	0.412	**0.503**	0.039	0.382	0.830	**0.843**	0.400	0.816	0.900	0.907	0.756	**0.915**
scottish-gaelic	0.449	**0.498**	0.004	0.441	0.441	**0.506**	0.087	0.490	–	–	–	–
serbo-croatian	0.285	0.291	0.001	**0.363**	0.570	0.601	0.095	**0.683**	0.863	0.850	0.166	**0.898**
slovak	0.647	**0.705**	0.006	0.447	0.720	**0.779**	0.295	0.659	0.777	**0.805**	0.530	0.789
slovene	0.616	**0.834**	0.000	0.583	0.767	**0.886**	0.352	0.834	0.798	**0.943**	0.636	0.933
spanish	0.787	0.882	0.000	**0.901**	0.911	0.895	0.192	**0.971**	0.954	0.908	0.717	**0.978**
swedish	0.421	**0.475**	0.049	0.208	0.635	**0.795**	0.282	0.643	0.723	**0.843**	0.583	0.789
turkish	0.124	0.624	0.000	**0.805**	0.613	0.876	0.303	**0.977**	0.825	0.921	0.697	**0.994**
ukrainian	0.523	**0.594**	0.007	0.411	**0.734**	0.709	0.285	0.655	**0.808**	0.760	0.452	0.773
urdu	0.670	0.670	0.010	**0.883**	0.680	0.680	0.027	**0.953**	**0.991**	0.488	0.221	0.982
welsh	0.601	0.349	0.000	**0.857**	0.693	0.864	0.127	**0.939**	0.752	0.903	0.258	**0.960**
basque	0.040	0.180	0.005	**0.890**	0.051	0.182	0.021	**0.891**	–	–	–	–
bengali	0.661	**0.928**	0.036	0.780	0.847	**0.963**	0.100	0.906	0.847	**0.965**	0.238	0.933
estonian	0.385	0.734	0.001	**0.806**	0.551	0.767	0.064	**0.953**	0.581	0.779	0.273	**0.951**
haida	0.554	0.810	0.000	**0.937**	0.802	0.849	0.002	**0.937**	–	–	–	–
irish	0.317	**0.439**	0.045	0.375	0.424	0.493	0.137	**0.592**	0.474	0.530	0.411	**0.692**
khaling	0.247	0.495	0.011	**0.973**	0.546	0.627	0.279	**0.992**	0.840	0.659	0.638	**0.996**
kurmanji	0.633	**0.648**	0.003	0.449	0.790	**0.798**	0.279	0.695	0.875	0.844	0.679	**0.878**
latin	0.336	**0.594**	0.000	0.157	0.449	**0.889**	0.112	0.691	0.493	**0.922**	0.301	0.820
lithuanian	0.536	**0.669**	0.006	0.487	0.615	**0.831**	0.059	0.744	0.662	**0.879**	0.302	0.876
norwegian-bokmal	0.417	**0.438**	0.396	0.306	**0.590**	**0.590**	0.576	0.340	**0.750**	**0.750**	0.715	0.667
romanian	0.151	**0.232**	0.008	0.062	0.630	**0.715**	0.077	0.561	0.773	**0.786**	0.284	0.744
sorani	0.534	0.532	0.000	**0.630**	0.661	0.561	0.065	**0.879**	0.646	0.599	0.488	**0.898**
Average:	0.520	**0.607**	0.035	0.588	0.682	0.748	0.220	**0.783**	0.783	0.802	0.549	**0.886**

Table 4: Accuracies for task 2. All systems are described in the text. Upper part: development languages; lower part: surprise languages.

"riding a bike" and "hike" $\mapsto$ "hiking" are not the same, even though they should be. Such cases potentially confuse the algorithm. A solution could be to detect training examples which consist of more than one token and split them up, in order to just consider the inflecting word.

Fine-tuning. For some settings and languages, e.g., Danish or Bokmål, the fine-tuned model obtains a lower accuracy than the base MED* model. While this might seem strange at first, when comparing to Table 1, it gets clear that this is mainly the case for languages where, besides the lemma,

no forms of a paradigm are given. This results in the model being fine-tuned on autoencoding the lemma, and thus a strong bias to copy the input, which can hurt performance. A possible solution might be to apply a combination of fine-tuning and multi-domain training as proposed, e.g., by Chu et al. (2017) for neural machine translation. We leave respective experiments for future work.

10 Conclusion

We presented the LMU system for the CoNLL-SIGMORPHON 2017 shared task on universal morphological reinflection, which is based on an encoder-decoder network. We introduced two new methods for handling multi-source morphological reinflection: CIS, a source form selection algorithm based on edit trees and a fine-tuning approach similar in spirit to domain adaptation. On average over all participating languages, our approaches outperform the official shared task baseline for both tasks and all settings.

Acknowledgments

We would like to thank VolkswagenStiftung for supporting this research.

References

Roee Aharoni, Yoav Goldberg, and Yonatan Belinkov. 2016. Improving sequence to sequence learning for morphological inflection generation: The biu-mit systems for the sigmorphon 2016 shared task for morphological reinflection. In *SIGMORPHON*.

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

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint 1409.1259* .

Kyunghyun Cho, Bart Van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMNLP*.

Grzegorz Chrupała. 2008. *Towards a machine-learning architecture for lexical functional grammar parsing*. Ph.D. thesis, Dublin City University.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of simple domain adaptation methods for neural machine translation. *arXiv preprint 1701.03214* .

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017a. The CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *CoNLL-SIGMORPHON*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *SIGMORPHON*.

Ryan Cotterell, John Sylak-Glassman, and Christo Kirov. 2017b. Neural graphical models over strings for principal parts morphological paradigm completion. In *EACL*.

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

Dan Gusfield. 1997. *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge university press.

Katharina Kann and Hinrich Schütze. 2016a. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *SIGMORPHON*.

Katharina Kann and Hinrich Schütze. 2016b. Single-model encoder-decoder with explicit morphological representation for reinflection. In *ACL*.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *CoRR* abs/1504.00941.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *IWSLT*.

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

Matthew D Zeiler. 2012. ADADELTA: An adaptive learning rate method. *CoRR* abs/1212.5701.

Chunting Zhou and Graham Neubig. 2017. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. *arXiv preprint 1704.01691* .

Chapter 8

Neural Morphological Analysis: Encoding-Decoding Canonical Segments

Neural Morphological Analysis: Encoding-Decoding Canonical Segments

Katharina Kann
Center for Information and Language Processing
LMU Munich, Germany
kann@cis.lmu.de

Ryan Cotterell
Department of Computer Science
Johns Hopkins University, USA
ryan.cotterell@jhu.edu

Hinrich Schütze
Center for Information and Language Processing
LMU Munich, Germany
inquiries@cislmu.org

Abstract

Canonical morphological segmentation aims to divide words into a sequence of standardized segments. In this work, we propose a character-based neural encoder-decoder model for this task. Additionally, we extend our model to include morpheme-level and lexical information through a neural reranker. We set the new state of the art for the task improving previous results by up to 21% accuracy. Our experiments cover three languages: English, German and Indonesian.

1 Introduction

Morphological segmentation aims to divide words into morphemes, meaning-bearing sub-word units. Indeed, segmentations have found use in a diverse set of NLP applications, e.g., automatic speech recognition (Afify et al., 2006), keyword spotting (Narasimhan et al., 2014), machine translation (Clifton and Sarkar, 2011) and parsing (Seeker and Çetinoğlu, 2015). In the literature, most research has traditionally focused on *surface segmentation*, whereby a word w is segmented into a sequence of substrings whose concatenation is the entire word; see Ruokolainen et al. (2016) for a survey. In contrast, we consider *canonical segmentation*: w is divided into a sequence of standardized segments. To make the difference concrete, consider the following example: the surface segmentation of the complex English word *achievability* is *achiev+abil+ity*, whereas its canonical segmentation is *achieve+able+ity*, i.e., we restore the alterations made during word formation.

Canonical versions of morphological segmentation have been introduced multiple times in the literature (Kay, 1977; Naradowsky and Goldwater, 2009; Cotterell et al., 2016). Canonical segmentation has several representational advantages over surface segmentation, e.g., whether two words share a morpheme is no longer obfuscated by orthography. However, it also introduces a hard algorithmic challenge: in addition to segmenting a word, we must reverse orthographic changes, e.g., mapping *achievability*$\mapsto$*achieveableity*.

Computationally, canonical segmentation can be seen as a sequence-to-sequence problem: we must map a word form to a canonicalized version with segmentation boundaries. Inspired by the recent success of neural encoder-decoder models (Sutskever et al., 2014) for sequence-to-sequence problems in NLP, we design a neural architecture for the task. However, a naïve application of the encoder-decoder model ignores much of the linguistic structure of canonical segmentation—it cannot directly model the individual canonical segments, e.g., it cannot easily produce segment-level embeddings. To solve this, we use a neural reranker on top of the encoder-decoder, allowing us to embed both characters and entire segments. The combined approach outperforms the state of the art by a wide margin (up to 21% accuracy) in three languages: English, German and Indonesian.

2 Neural Canonical Segmentation

We begin by formally describing the canonical segmentation task. Given a discrete alphabet Σ (e.g., the 26 letters of the English alphabet),

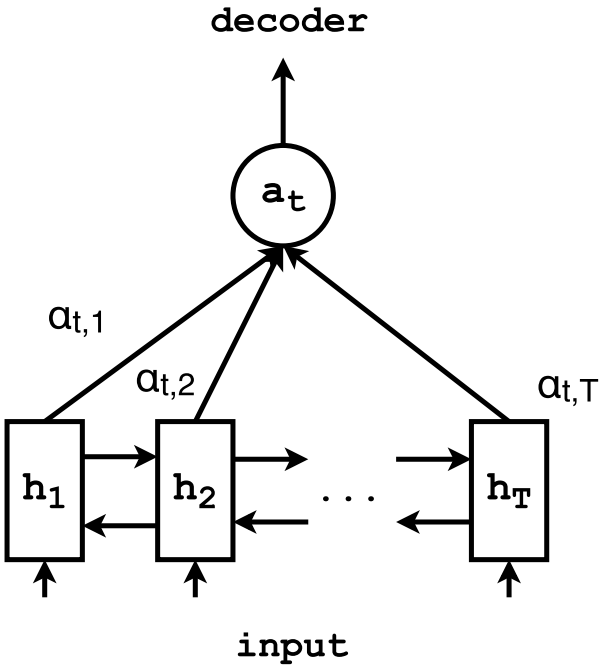


Figure 1: Detailed view of the attention mechanism of the neural encoder-decoder.

our goal is to map a word $w \in \Sigma^*$ (e.g., w=*achievability*), to a canonical segmentation $c \in \Omega^*$ (e.g., c=*achieve+able+ity*). We define $\Omega = \Sigma \cup \{+\}$, where $+$ is a distinguished separation symbol. Additionally, we will write the segmented form as $c=\sigma_1+\sigma_2+\ldots+\sigma_n$, where each segment $\sigma_i \in \Sigma^*$ and n is the number of canonical segments.

We take a probabilistic approach and, thus, attempt to learn a distribution $p(c \mid w)$. Our model consists of two parts. First, we apply an encoder-decoder recurrent neural network (RNN) (Bahdanau et al., 2014) to the sequence of characters of the input word to obtain candidate canonical segmentations. Second, we define a neural reranker that allows us to embed individual morphemes and chooses the final answer from within a set of candidates generated by the encoder-decoder.

2.1 Neural Encoder-Decoder

Our encoder-decoder is based on Bahdanau et al. (2014)'s neural machine translation model.[1] The **encoder** is a bidirectional gated RNN (GRU) (Cho et al., 2014b). Given a word $w \in \Sigma^*$, the input to the encoder is the sequence of characters of w, represented as one-hot vectors. The **decoder** defines a conditional probability distribution over $c \in \Omega^*$ given w:

$$\begin{aligned} p_{\text{ED}}(c \,|\, w) &= \prod_{t=1}^{|c|} p(c_t | c_1, \ldots, c_{t-1}, w) \\ &= \prod_{t=1}^{|c|} g(c_{t-1}, s_t, a_t) \end{aligned}$$

where g is a nonlinear activation function, s_t is the state of the decoder at t and a_t is a weighted sum of the $|w|$ states of the encoder. The state of the encoder for w_i is the concatenation of forward and backward hidden states $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ for w_i. An overview of how the attention weight and the weighted sum a_t are included in the architecture can be seen in Figure 1. The attention weights $\alpha_{t,i}$ at each timestep t are computed based on the respective encoder state and the decoder state s_t. See Bahdanau et al. (2014) for further details.

2.2 Neural Reranker

The encoder-decoder, while effective, predicts each output character in Ω sequentially. It does not use explicit representations for entire segments and is incapable of incorporating simple lexical information, e.g., does this canonical segment occur as an independent word in the lexicon? Therefore, we extend our model with a reranker.

The reranker rescores canonical segmentations from a candidate set, which in our setting is sampled from p_{ED}. Let the sample set be $\mathcal{S}_w = \{k^{(i)}\}_{i=1}^{N}$ where $k^{(i)} \sim p_{\text{ED}}(c \mid w)$. We define the neural reranker as

$$p_\theta(c \,|\, w) = \frac{\exp\left(u^\top \tanh(W v_c) + \tau \log p_{\text{ED}}(c|w)\right)}{Z_\theta}$$

where $v_c = \sum_{i=1}^{n} v_{\sigma_i}$ (recall $c = \sigma_1+\sigma_2+\ldots+\sigma_n$) and v_{σ_i} is a one-hot morpheme embedding of σ_i with an additional binary dimension marking if σ_i occurs independently as a word in the language.[2] The partition function is $Z_\theta(w)$ and the parameters are $\theta = \{u, W, \tau\}$. The parameters W and u

[1] github.com/mila-udem/blocks-examples/tree/master/machine_translation

[2] To determine if a canonical segment is in the lexicon, we check its occurrence in ASPELL. Alternatively, one could ask whether it occurs in a large corpus, e.g., Wikipedia.

are projection and hidden layers, respectively, of a multi-layered perceptron and τ can be seen as a temperature parameter that anneals the encoder-decoder model p_{ED} (Kirkpatrick, 1984). We define the partition function over the sample set $\mathcal{S}_w$:

$$Z_\theta = \sum_{k \in \mathcal{S}_w} \exp\left(u^\top \tanh(W v_k) + \tau \log p_{\text{ED}}(k \mid w)\right).$$

The reranking model's ability to embed morphemes is important for morphological segmentation since we often have strong corpus-level signals. The reranker also takes into account the character-level information through the score of the encoder-decoder model. Due to this combination we expect stronger performance.

3 Related Work

Various approaches to morphological segmentation have been proposed in the literature. In the unsupervised realm, most work has been based on the principle of minimum description length (Cover and Thomas, 2012), e.g., LINGUISTICA (Goldsmith, 2001; Lee and Goldsmith, 2016) or MORFESSOR (Creutz and Lagus, 2002; Creutz et al., 2007; Poon et al., 2009). MORFESSOR was later extended to a semi-supervised version by Kohonen et al. (2010). Supervised approaches have also been considered. Most notably, Ruokolainen et al. (2013) developed a supervised approach for morphological segmentation based on conditional random fields (CRFs) which they later extended to work also in a semi-supervised way (Ruokolainen et al., 2014) using letter successor variety features (Hafer and Weiss, 1974). Similarly, Cotterell et al. (2015) improved performance with a semi-Markov CRF.

More recently, Wang et al. (2016) achieved state-of-the-art results on surface morphological segmentation using a window LSTM. Even though Wang et al. (2016) also employ a recurrent neural network, we distinguish our approach, in that we focus on *canonical* morphological segmentation, rather than *surface* morphological segmentation.

Naturally, our approach is also relevant to other applications of recurrent neural network transduction models (Sutskever et al., 2014; Cho et al., 2014a). In addition to machine translation (Bahdanau et al., 2014), these models have been successfully applied to many areas of NLP, including parsing (Vinyals et al., 2015), morphological reinflection (Kann and Schütze, 2016) and automatic speech recognition (Graves and Schmidhuber, 2005; Graves et al., 2013).

4 Experiments

To enable comparison to earlier work, we use a dataset that was prepared by Cotterell et al. (2016) for canonical segmentation.[3]

4.1 Languages

The dataset we work on covers 3 languages: English, German and Indonesian. English and German are West Germanic Languages, with the former being an official languages in nearly 60 different states and the latter being mainly spoken in Western Europe. Indonesian — or *Bahasa Indonesia*— is the official language of Indonesia.

Cotterell et al. (2016) report the best experimental results for Indonesian, followed by English and finally German. The high error rate for German might be caused by it being rich in orthografic changes. In contrast, Indonesian morphology is comparatively simple.

4.2 Corpora

The data for the English language was extracted from segmentations derived from the CELEX database (Baayen et al., 1993). The German data was extracted from DerivBase (Zeller et al., 2013), which provides a collection of derived forms together with the transformation rules, which were used to create the canonical segmentations. Finally, the data for Bahasa Indonesia was collected by using the output of the MORPHIND analyzer (Larasati et al., 2011), together with an open-source corpus of Indonesian. For each language we used the 10,000 forms that were selected at random by Cotterell et al. (2016) from a uniform distribution over types to form the corpus. Following them, we perform our experiments on 5 splits of the data into 8000 training forms, 1000 development forms and 1000 test forms and report averages.

[3] ryancotterell.github.io/canonical-segmentation

4.3 Training

We train an ensemble of five encoder-decoder models. The encoder and decoder RNNs each have 100 hidden units. Embedding size is 300. We use ADADELTA (Zeiler, 2012) with a minibatch size of 20. We initialize all weights (encoder, decoder, embeddings) to the identity matrix and the biases to zero (Le et al., 2015). All models are trained for 20 epochs. The hyperparameter values are taken from Kann and Schütze (2016) and kept unchanged for the application to canonical segmentation described here.

To train the reranking model, we first gather the sample set $\mathcal{S}_w$ on the training data. We take 500 individual samples, but (as we often sample the same form multiple times) $|\mathcal{S}_w| \approx 5$. We optimize the log-likelihood of the training data using ADADELTA. For generalization, we employ L_2 regularization and we perform grid search to determine the coefficient $\lambda \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. To decode the model, we again take 500 samples to populate $\mathcal{S}_w$ and select the best segmentation.

Baselines. Our first baseline is the joint transduction and segmentation model (**JOINT**) of Cotterell et al. (2016). It is the current state of the art on the datasets we use and the task of canonical segmentation in general. This model uses a jointly trained, separate transduction and segmentation component. Importantly, the joint model of Cotterell et al. (2016) *already contains segment-level features*. Thus, reranking this baseline would not provide a similar boost.

Our second baseline is a weighted finite-state transducer (**WFST**) (Mohri et al., 2002) with a log-linear parameterization (Dreyer et al., 2008), again, taken from Cotterell et al. (2016). The WFST baseline is particularly relevant because, like our encoder-decoder, it formulates the problem directly as a string-to-string transduction.

Evaluation Metrics. We follow Cotterell et al. (2016) and use the following evaluation measures: error rate, edit distance and morpheme F_1. Error rate is defined as 1 minus the proportion of guesses that are completely correct. Edit distance is the Levenshtein distance between guess and gold standard. For this, guess and gold are each represented as one string with a distinguished character denoting the segment boundaries. Morpheme F_1 compares the morphemes in guess and gold. Precision (resp. recall) is the proportion of morphemes in guess (resp. gold) that occur in gold (resp. guess).

		RR	ED	Joint	WFST	UB
error	en	**.19** (.01)	.25 (.01)	0.27 (.02)	0.63 (.01)	.06 (.01)
	de	**.20** (.01)	.26 (.02)	0.41 (.03)	0.74 (.01)	.04 (.01)
	id	**.05** (.01)	.09 (.01)	0.10 (.01)	0.71 (.01)	.02 (.01)
edit	en	**.21** (.02)	.47 (.02)	0.98 (.34)	1.35 (.01)	.10 (.02)
	de	**.29** (.02)	.51 (.03)	1.01 (.07)	4.24 (.20)	.06 (.01)
	id	**.05** (.00)	.12 (.01)	0.15 (.02)	2.13 (.01)	.02 (.01)
F_1	en	**.82** (.01)	.78 (.01)	0.76 (.02)	0.53 (.02)	.96 (.01)
	de	**.87** (.01)	.86 (.01)	0.76 (.02)	0.59 (.02)	.98 (.00)
	id	**.96** (.01)	.93 (.01)	0.80 (.01)	0.62 (.02)	.99 (.00)

Table 1: Error rate (top), edit distance (middle), F_1 (bottom) for canonical segmentation. Each double column gives the measure and its standard deviation. Best result on each line (excluding UB) in bold. RR: encoder-decoder+reranker. ED: encoder-decoder. JOINT, WFST: baselines (see text). UB: upper bound, the maximum score our reranker could obtain, i.e., considering the best sample in the predictions of ED.

5 Results

The results of the canonical segmentation experiment in Table 1 show that both of our models improve over all baselines. The encoder-decoder alone has a .02 (English), .15 (German) and .01 (Indonesion) lower error rate than the best baseline. The encoder-decoder improves most for the language for which the baselines did worst. This suggests that, for more complex languages, a neural network model might be a good choice.

The reranker achieves an additional improvement of .04 to .06. for the error rate. This is likely due to the additional information the reranker has access to: morpheme embeddings and existing words.

Important is also the upper bound we report. It shows the maximum performance the reranker could achieve, i.e., evaluates the best solution that appears in the set of candidate answers for the reranker. The right answer is contained in $\geq 94\%$ of samples. Note that, even though the upper bound goes up with the number of samples we take, there is no guarantee for any finite number of samples that they will contain the true answer. Thus, we would need to take an infinite number of samples to get a perfect upper bound. However, as the current upper bound is quite high, the encoder-decoder proves to be an appropri-

ate model for the task. Due to the large gap between the performance of the encoder-decoder and the upper bound, a better reranker could further increase performance. We will investigate ways to improve the reranker in future work.

Error analysis. We give for representative samples the error (E for the segmentation produced by our method) and the correct analysis (G for gold).

We first analyze cases in which the right answer does not appear at all in the samples drawn from the encoder-decoder. Those include problems with umlauts in German (G: *verflüchtigen*↦ *ver+flüchten+ig*, E: *verflucht+ig*) and orthographic changes at morpheme boundaries (G:*cutter*↦*cut+er*, E: *cutter* or *cutt+er*, sampled with similar frequency). There are also errors that are due to problems with the annotation, e.g., the following two gold segmentations are arguably incorrect: *tec*↦*detective* and *syrerin*↦*syr+er+in* (*syr* is neither a word nor an affix in German).

In other cases, the encoder-decoder does find the right solution (G), but gives a higher probability to an incorrect analysis (E). Examples are a wrong split into adjectives or nouns instead of verbs (G: *fügsamkeit*↦*fügen+sam+keit*, E: *fügsam+keit*), the other way around (G: *zähler*↦*zahl+er*, E: *zählen+er*), cases where the wrong morphemes are chosen (G: *precognition*↦*pre+cognition*, E: *precognit+ion*), difficult cases where letters have to be inserted (G: *redolence*↦*redolent+ence*, E: *re+dolence*) or words the model does not split up, even though they should be (G: *additive*↦ *addition+ive*, E: *additive*).

Based on its access to lexical information and morpheme embeddings, the reranker is able to correct some of the errors made by the encoder-decoder. Samples are G: *geschwisterpärchen*↦ *geschwisterpaar+chen*, E: *geschwisterpar+chen* (*geschwisterpaar* is a word in German but *geschwisterpar* is not) or G: *zickig*↦ *zicken+ig*, E: *zick+ig* (with *zicken*, but not *zick*, being a German word).

Finally, we want to know if segments that appear in the test set without being present in the training set are a source of errors. In order to investigate that, we split the test samples into two groups: The first group contains the samples for which our system finds the right answer. The second one contains all other samples. We compare the percentage of samples that do not appear in the training data for both groups. We exemplarily use the German data and the results results are shown in Table 2. First, it can be seen that very roughly about a third of all segments does not appear in the training data. This is mainly due to unseen lemmas as their stems are naturally unknown to the system. However, the correctly solved samples contain nearly 10% more unseen segments. As the average number of segments per word for wrong and right solutions — 2.44 and 2.11, respectively — does not differ by much, it seems unlikely that many errors are caused by unknown segments.

wrong samples	right samples
27.33 (.02)	36.60 (.01)

Table 2: Percentage of segments in the solutions for the test data that do not appear in the training set - split by samples that our system does or does not get right. We use the German data and average over the 5 splits. Standard deviation in parenthesis.

6 Conclusion and Future Work

We developed a model consisting of an encoder-decoder and a neural reranker for the task of canonical morphological segmentation. Our model combines character-level information with features on the morpheme level and external information about words. It defines a new state of the art, improving over baseline models by up to .21 accuracy, 16 points F_1 and .77 Levenshtein distance.

We found that $\geq 94\%$ of correct segmentations are in the sample set drawn from the encoder-decoder model, demonstrating the upper bound on the performance of our reranker is quite high; in future work, we hope to develop models to exploit this.

Acknowledgments

We gratefully acknowledge the financial support of Siemens for this research.

References

Mohamed Afify, Ruhi Sarikaya, Hong-Kwang Jeff Kuo, Laurent Besacier, and Yuqing Gao. 2006. On the use of morphological analysis for dialectal Arabic speech recognition. In *Proc. of INTERSPEECH*.

R. H. Baayen, R. Piepenbrock, and H. Van Rijn. 1993. The CELEX lexical data base on CD-ROM.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Kyunghyun Cho, Bart Van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. of EMNLP*.

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

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015. Labeled morphological segmentation with semi-markov models. In *Proc. of CoNLL*.

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

Thomas M Cover and Joy A Thomas. 2012. *Elements of Information Theory*. John Wiley & Sons.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proc. of the ACL-02 Workshop on Morphological and Phonological Learning*.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. 2007. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Transactions on Speech and Language Processing*, 5(1):3:1–3:29.

Markus Dreyer, Jason R. Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proc. of EMNLP*.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proc of. ICASSP*.

Margaret A. Hafer and Stephen F. Weiss. 1974. Word segmentation by letter successor varieties. *Information storage and retrieval*, 10(11):371–385.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proc. of ACL*.

Martin Kay. 1977. Morphological and syntactic analysis. *Linguistic Structures Processing*, 5:131–234.

Scott Kirkpatrick. 1984. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5-6):975–986.

Oskar Kohonen, Sami Virpioja, and Krista Lagus. 2010. Semi-supervised learning of concatenative morphology. In *Proc. of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*.

Septina Dian Larasati, Vladislav Kuboň, and Daniel Zeman. 2011. Indonesian morphology tool (morphind): Towards an indonesian corpus. In *Proc. of SFCM*. Springer.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Jackson L. Lee and John A. Goldsmith. 2016. Linguistica 5: Unsupervised learning of linguistic structure. In *Proc. of NAACL*.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

Jason Naradowsky and Sharon Goldwater. 2009. Improving morphology induction by learning spelling rules. In *Proc. of IJCAI*.

Karthik Narasimhan, Damianos Karakos, Richard Schwartz, Stavros Tsakalidis, and Regina Barzilay. 2014. Morphological segmentation for keyword spotting. In *Proc. of EMNLP*.

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

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *Proc. of CoNLL*.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and mikko kurimo. 2014. Painless semi-supervised morphological segmentation using conditional random fields. In *Proc. of EACL*.

Teemu Ruokolainen, Oskar Kohonen, Kairit Sirts, Stig-Arne Grönroos, Mikko Kurimo, and Sami Virpioja. 2016. Comparative study of minimally supervised morphological segmentation. *Computational Linguistics*, 42(1):91–120.

Wolfgang Seeker and Özlem Çetinoğlu. 2015. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *TACL*, 3:359–373.

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

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proc. of NIPS*.

Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. 2016. Morphological segmentation with window LSTM neural networks. In *Proc. of AAAI*.

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

Britta Zeller, Jan Šnajder, and Sebastian Padó. 2013. Derivbase: Inducing and evaluating a derivational morphology resource for german. In *Proc. of ACL*.

Bibliography

Mohamed Afify, Ruhi Sarikaya, Hong-Kwang Jeff Kuo, Laurent Besacier, and Yuqing Gao. On the use of morphological analysis for dialectal Arabic speech recognition. In *Annual Conference of the International Speech Communication Association*, 2006.

Roee Aharoni and Yoav Goldberg. Sequence to sequence transduction with hard monotonic attention. In *Annual Meeting of the Association for Computational Linguistics*, 2017.

Roee Aharoni, Yoav Goldberg, and Yonatan Belinkov. Improving sequence to sequence learning for morphological inflection generation: The BIU-MIT systems for the SIGMORPHON 2016 shared task for morphological reinflection. In *SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 2016.

Malin Ahlberg, Markus Forsberg, and Mans Hulden. Semi-supervised learning of morphological paradigms and lexicons. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2014.

Malin Ahlberg, Markus Forsberg, and Mans Hulden. Paradigm classification in supervised learning of morphology. In *Conference of the North American Chapter of the Association for Computational Linguistics / Human Language Technologies*, 2015.

Inaki Alegria and Izaskun Etxeberria. EHU at the SIGMORPHON 2016 shared task. A simple proposal: Grapheme-to-phoneme for inflection. In *SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 2016.

Oded Avraham and Yoav Goldberg. The interplay of semantics and morphology in word embeddings. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2017.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. What do neural machine translation models learn about morphology? *Annual Meeting of the Association for Computational Linguistics*, 2017.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. Training data augmentation for low-resource morphological inflection. In *CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, 2017.

Tim Buckwalter. Buckwalter Arabic morphological analyzer version 2.0. Technical report, Linguistic Data Consortium, University of Pennsylvania, 2004.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014a.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, 2014b.

Ann Clifton and Anoop Sarkar. Combining morpheme-based machine translation with post-processing morpheme prediction. In *Annual Meeting of the Association for Computational Linguistics*, 2011.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. Labeled morphological segmentation with semi-markov models. In *Computational Natural Language Learning*, 2015.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. The SIGMORPHON 2016 shared task—morphological reinflection. In *SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 2016a.

Ryan Cotterell, Tim Vieira, and Hinrich Schütze. A joint model of orthography and morphological segmentation. In *North American Chapter of the Association for Computational Linguistics*, 2016b.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. The CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, 2017a.

Ryan Cotterell, John Sylak-Glassman, and Christo Kirov. Neural graphical models over strings for principal parts morphological paradigm completion. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2017b.

Mathias Creutz and Krista Lagus. Unsupervised discovery of morphemes. In *Workshop on Morphological and Phonological Learning*, 2002.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Transactions on Speech and Language Processing*, 5(1):3:1–3:29, 2007.

Adrià De Gispert, Sami Virpioja, Mikko Kurimo, and William Byrne. Minimum Bayes risk combination of translation hypotheses from alternative morphological decompositions. In *North American Chapter of the Association for Computational Linguistics*, pages 73–76, 2009.

Aliya Deri and Kevin Knight. How to make a frenemy: Multitape FSTs for portmanteau generation. In *Conference of the North American Chapter of the Association for Computational Linguistics / Human Language Technologies*, 2015.

Markus Dreyer, Jason R. Smith, and Jason Eisner. Latent-variable modeling of string transductions with finite-state methods. In *Conference on Empirical Methods in Natural Language Processing*, 2008.

Greg Durrett and John DeNero. Supervised learning of complete morphological paradigms. In *Conference of the North American Chapter of the Association for Computational Linguistics / Human Language Technologies*, 2013.

Jeffrey L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Ramy Eskander, Nizar Habash, and Owen Rambow. Automatic extraction of morphological lexicons from morphologically annotated corpora. In *Conference on Empirical Methods in Natural Language Processing*, 2013.

Manaal Faruqui, Ryan McDonald, and Radu Soricut. Morpho-syntactic lexicon generation using graph-based semi-supervised learning. *Transactions of the Association for Computational Linguistics*, 4:1–16, 2016a.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. Morphological inflection generation using character sequence to sequence learning. In *Conference of the North American Chapter of the Association for Computational Linguistics / Human Language Technologies*, 2016b.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Conference of the North American Chapter of the Association for Computational Linguistics / Human Language Technologies*, 2016.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, 2017.

John Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198, 2001.

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

Spence Green and John DeNero. A class-based agreement model for generating accurately inflected translations. In *Annual Meeting of the Association for Computational Linguistics*, 2012.

Margaret A. Hafer and Stephen F. Weiss. Word segmentation by letter successor varieties. *Information storage and retrieval*, 10(11):371–385, 1974.

John A.. Hertz, Anders S. Krogh, and Richard G. Palmer. *Introduction to the theory of neural computation*. Addison-Wesley Publishing Company, 1991.

Teemu Hirsimäki, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja, and Janne Pylkkönen. Unlimited vocabulary speech recognition with morph language models applied to finnish. *Computer Speech & Language*, 20(4): 515–541, 2006.

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

Mans Hulden, Markus Forsberg, and Malin Ahlberg. Semi-supervised learning of morphological paradigms and lexicons. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2014.

Klara Janecki. *300 Polish Verbs*. Barron's Educational Series, 2000.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.

Katharina Kann and Hinrich Schütze. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 2016.

Martin Kay. Morphological and syntactic analysis. *Linguistic Structures Processing*, 5:131–234, 1977.

David L. King. Evaluating sequence alignment for learning inflectional morphology. In *SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 2016.

Philipp Koehn and Kevin Knight. Empirical methods for compound splitting. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2003.

Oskar Kohonen, Sami Virpioja, and Krista Lagus. Semi-supervised learning of concatenative morphology. In *SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 2010.

Kimmo Koskenniemi. Two-level model for morphological analysis. In *International Joint Conference on Artificial Intelligence*, volume 83, pages 683–685, 1983.

Jackson L. Lee and John A. Goldsmith. Linguistica 5: Unsupervised learning of linguistic structure. In *North American Chapter of the Association for Computational Linguistics*, 2016.

Jindřich Libovickỳ and Jindřich Helcl. Attention strategies for multi-source sequence-to-sequence learning. In *Annual Meeting of the Association for Computational Linguistics*, 2017.

Ling Liu and Lingshuang Jack Mao. Morphological reinflection with conditional random fields and unsupervised features. In *SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 2016.

Klaus Macherey, Andrew M. Dai, David Talbot, Ashok C. Popat, and Franz Och. Language-independent compound splitting with morphological operations. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. Align and copy: UZH at SIGMORPHON 2017 shared task for morphological reinflection. In *CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, 2017.

Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biology*, 5(4):115–133, 1943.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. Efficient higher-order CRFs for morphological tagging. In *Conference on Empirical Methods in Natural Language Processing*, 2013.

Jason Naradowsky and Sharon Goldwater. Improving morphology induction by learning spelling rules. In *International Joint Conference on Artificial Intelligence*, 2009.

Karthik Narasimhan, Damianos Karakos, Richard Schwartz, Stavros Tsakalidis, and Regina Barzilay. Morphological segmentation for keyword spotting. In *Conference on Empirical Methods in Natural Language Processing*, 2014.

Garrett Nicolai and Grzegorz Kondrak. Morphological analysis without expert annotation. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2017.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. Inflection generation as discriminative string transduction. In *Conference of the North American Chapter of the Association for Computational Linguistics / Human Language Technologies*, 2015.

BIBLIOGRAPHY

Garrett Nicolai, Bradley Hauer, Adam St Arnaud, and Grzegorz Kondrak. Morphological reinflection via discriminative string transduction. In *SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 2016.

Robert Östling. Morphological reinflection with convolutional neural networks. In *SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 2016.

Jorma Rissanen. Stochastic complexity in statistical inquiry. *World scientific series in computer science*, 15:79–93, 1989.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *Computational Natural Language Learning*, 2013.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. Painless semi-supervised morphological segmentation using conditional random fields. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2014.

Teemu Ruokolainen, Oskar Kohonen, Kairit Sirts, Stig-Arne Grönroos, Mikko Kurimo, and Sami Virpioja. Comparative study of minimally supervised morphological segmentation. *Computational Linguistics*, 42(1):91–120, 2016.

Tarek Sakakini, Suma Bhat, and Pramod Viswanath. Fixing the infix: Unsupervised discovery of root-and-pattern morphology. *arXiv preprint arXiv:1702.02211*, 2017.

Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, 2005.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. SMOR: A german computational morphology covering derivation, composition and inflection. In *International Conference on Language Resources and Evaluation*, 2004.

Wolfgang Seeker and Özlem Çetinoğlu. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *Transactions of the Association for Computational Linguistics*, 3:359–373, 2015.

Alexey Sorokin. Using longest common subsequence and character models to predict word forms. In *SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 2016.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

Dima Taji, Ramy Eskander, Nizar Habash, and Owen Rambow. The Columbia University-New York University Abu Dhabi SIGMORPHON 2016 morphological reinflection shared task submission. In *SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, 2015.

Ekaterina Vylomova, Ryan Cotterell, Timothy Baldwin, and Trevor Cohn. Context-aware prediction of derivational word-forms. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2017.

Terence Wade. *A comprehensive Russian grammar*. John Wiley & Sons, 2010.

Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. Morphological segmentation with window LSTM neural networks. In *AAAI Conference on Artificial Intelligence*, 2016.

Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. *Neurocomputing: Foundations of Research*, 4:96–104, 1960.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y. Ng. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*, 2016.

Chunting Zhou and Graham Neubig. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *Annual Meeting of the Association for Computational Linguistics*, 2017.

Barret Zoph and Kevin Knight. Multi-source neural translation. In *Conference of the North American Chapter of the Association for Computational Linguistics / Human Language Technologies*, 2016.