Computational Morphology with Neural Network Approaches

Ling Liu
University of Colorado
ling.liu@colorado.edu

Abstract

Neural network approaches have been applied to computational morphology with great success, improving the performance of most tasks by a large margin and providing new perspectives for modeling. This paper starts with a brief introduction to computational morphology, followed by a review of recent work on computational morphology with neural network approaches, to provide an overview of the area. In the end, we will analyze the advantages and problems of neural network approaches to computational morphology, and point out some directions to be explored by future research and study.

1 Introduction

In the word $drivers$, there are three parts which can't be split further without resulting in meaningless units in English: $drive$, -$(e)r$, and -s.[1] The first part $drive$ is a verb, meaning to operate to make vehicles move; the second part -$(e)r$ has the function of changing the verb to a noun meaning the person who does the actions; and the third part -s has the function of indicating there are more than one of this noun. Analyzing the internal structure of words, understanding the meaning and function related to each part, and figuring out how different units can be combined to make valid words, is the focus of the area of linguistic study known as morphology. According to Haspelmath and Sims (2013), "Morphology is the study of systematic covariation in the form and meaning of words." (p2) Note that this definition and the example in the beginning presupposes that language can be divided into distinct units of word. This paper focuses on studies which follow this presupposition and use data clearly marked with word boundaries. In other words, we focus on data where larger units have been segmented into words either with natural word boundaries like spaces in English or with some preprocessing for languages without natural word boundaries like Chinese and Japanese.

In the example of $drivers$, the three parts are three different morphemes, the first morpheme $drive$ is a word in English which can't the divided further, also called a stem, the other two parts which are not words but can be attached to stems or larger units are named affixes. There are two types of processes involved to form the word *drivers*: derivation and inflection. The process of concatenating $drive$ and -$(e)r$ to form $driver$ is a derivational process, where the new word has a different concrete lexical meaning; the process of adding -s to the end of $driver$ to get the word form *drivers* is more related to the grammatical requirements by the English language, and this process is known as inflection.

[1] "-" is used to indicate that the string is not a word by itself, -*string* indicates the string is usually added to the end of a word, i.e. it's a suffix; *string*- indicates the string is usually added to the beginning of a word, i.e. it's a prefix; and -*string*- indicates the string is usually inserted into a word, i.e. it's an infix.

Computational methods can be applied to facilitate morphological study, which is the focus of computational morphology. For example, finite state machines and two-level morphology provide ways to formulate the morphological rules linguists come up with to generalize over word formation in different languages and to use these rules to automatically analyze or generate more data. Finite state machines and two-level morphology are very linguistics-oriented and rely directly on linguistic studies. Machine learning techniques like Hidden Markov Models (HMM), Conditional Random Fields (CRFs), Support Vector Machines (SVMs) do not rely on linguistics as much. These machine learning techniques provide ways for the algorithm to learn about morphology from annotated data or raw text without explicit linguistic rules, though they still require high-level feature engineering and usually use heuristics designed with heavy linguistic considerations. Neural network models have been flourishing in recent years and generated state-of-the-art results for many tasks in natural language processing (NLP), including computational morphology. Since Kann and Schütze (2016a) applied the neural encoder-decoder models to SIGMORPHON 2016 shared task on morphological (re)inflection (Cotterell et al., 2016a), neural network models have been dominant in computational morphology and achieved significant improvements over previous results. Such models rely heavily on large amounts of annotated data in order to achieve good performance, but they don't require explicit linguistic rules or high-level feature engineering. Being free from feature engineering is an advantage from the engineering side, but is a disadvantage from the linguistics side because it's hard to interpret what the models learn, making it less helpful to contribute to the understanding of human languages.

From the NLP perspective, computational morphology has been an important data preprocessing step for downstream tasks like machine translation, information retrieval, dependency parsing etc. Though various ways to make use of subword information like characters, n-grams, or byte pair encodings have been explored, high-quality morphological processing remains most helpful, especially for morphologically rich languages (MRLs)[2] (Belinkov et al., 2017; Vania et al., 2018; Dehouck and Denis, 2018; Klein and Tsarfaty, 2020).

The overwhelmingly good performance of neural network models indicates that they are good models for NLP purposes, and they can't be ignored either for linguistic and cognitive understanding of language. With such a consideration, this paper will review recent work (mainly work published between 2016 and 2020) using **language-agnostic** neural network models[3] to process computational morphology **with the focus on inflection**. We will summarize what tasks have been using such models, what architectures the neural models have and what results have been generated. Then we will analyze the advantages and disadvantages of the neural models for the tasks, and point out some directions for future work.

The remaining of the paper is organized as follows: Section 2 will define tasks in computational morphology. Section 3 will introduce basic neural network architectures, which have been combined with each other or other machine learning approaches to process computational morphology. Sections 4 and 5 will explain in more detail how different models have been applied to different tasks. Section 6 will discuss the advantages and problems of neural network approaches, and point out some directions for future research and study. This paper will end with a conclusion in Section 7.

[2]Conforti et al. (2018) define MRL as "a language where word shapes encode a consistent number of syntactic and semantic features", like fusional languages such as Czech, and agglutinating languages such as Finnish, Hungarian, and Turkish.

[3]Language-agnostic models refer to models which have been developed for and tested on multiple languages without being engineered for each language specifically.

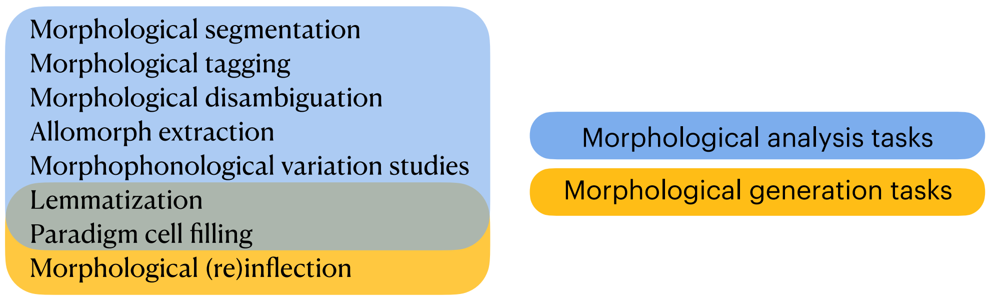

Morphological segmentation
Morphological tagging
Morphological disambiguation
Allomorph extraction
Morphophonological variation studies
Lemmatization
Paradigm cell filling
Morphological (re)inflection
Morphological analysis tasks
Morphological generation tasks

Figure 1: Summary of morphological learning tasks. Analysis tasks are shaded in blue (the top box), and generation tasks are shaded in orange (the bottom box). Lemmatization and paradigm cell filling (the overlap part between the top and bottom boxes) involve both analysis and generation.

2 Tasks in computational morphology

Tasks in computational morphology (summarized in Figure 1) can be generally classified into two types: analysis and generation. For morphological analysis, the goal is to learn about the morphological structure of a given word form. Analysis tasks include morphological tagging, morphological segmentation, morphological disambiguation, allomorphy extraction and morphophonological variation studies. For morphological generation, the goal is to generate a correct word form, including tasks like inflection, reinflection etc. When the reinflection is to generate the lemma form (i.e. the base or dictionary form) of a word given its other inflected forms, the task is called lemmatization. However, lemmatization is also closely related to morphological analysis: morphological tagging and morphological segmentation are usually indispensable from lemmatization. Paradigm completion tasks may also involve both analysis and generation processes. When the task is conducted on the lexicon, i.e. word in isolation, we call it a **type-based** task; and when the task is about words in context, usually sentential context, we call it a **token-based** task.

2.1 Morphological analysis

The term *morphological analysis* has been used quite generally to refer to all the morphological learning tasks to analyze or understand the given word form. This term has also been used in a more specific sense, where it refers to the task of "[annotating] a given word form with its lemma and morphological tag" (Nicolai and Kondrak, 2017) (p211), i.e. the combination of lemmatization and morphological tagging. In this paper, we use *morphological analysis* to refer to these two broader and narrower senses. In the literature, *morphological analysis* has also been used interchangeably with *morphological tagging* to refer to the task of "predicting fine-grained annotations about the syntactic properties of tokens in a language such as part-of-speech, case, or tense" (Malaviya et al., 2018) (p2653), but this paper avoids this usage.

2.1.1 Morphological tagging and lemmatization

Morphological tagging is closely related to part-of-speech (POS) tagging, and has been treated as fine-grained POS tagging (Labeau et al., 2015; Conforti et al., 2018) or a generalization of POS tagging (Cotterell and Heigold, 2017). A tag for morphological description usually consists of a POS label together with more specific labels which describe the morphosyntactic features of each

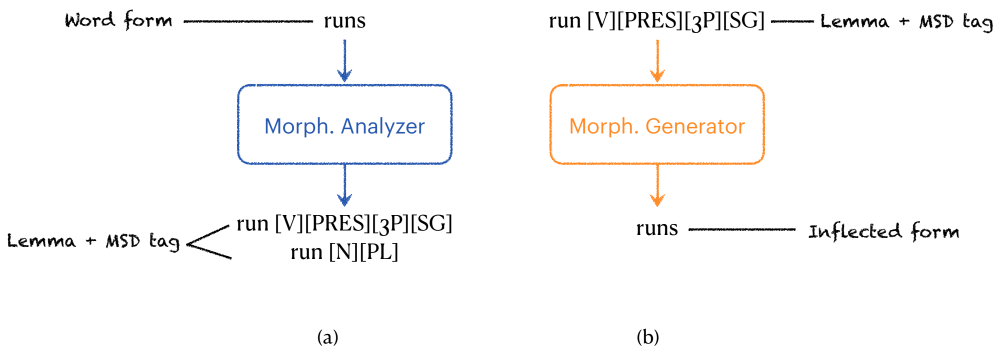


Figure 2: (a) is a morphological analyzer which conducts lemmatization and morphological tagging on the English word *runs*. (b) is a morphological generator which produces the inflected form corresponding to the English lemma *run* and the MSD tag *V;PRES;3P;SG*.

INPUT	Sentence	He	runs	a	investment	company	.
OUTPUT	Lemmatization	he	run	a	investment	company	.
OUTPUT	Morph. Tagging	PRON;NOM;SG	V;SG;3;PRS	DET;IND	N;SG	N;SG	PUNCT

Table 1: Example of lemmatization and morphological tagging in context.

word. Such morphological tags have been referred to as MSDs (morphosyntactic descriptions), features, labels, or tags. In this remainder of the paper, we will use **MSD tag** to refer to the whole chunk of labels corresponding to each word, e.g. V;PRES;3P;SG for *runs*, and refer to each component label like V, PRES, 3P, SG as its **features**. *Lemmatization* is the task to convert the word to the normalized form (i.e. lemma form), which is usually the base or dictionary form of the word (Plisson et al., 2004; Bergmanis and Goldwater, 2018). Figure 2(a)[4] illustrates a typical type-based morphological analyzer which conducts *lemmatization* and *morphological tagging* where the input is a word form and the morphological analyzer is expected to produce all the possible lemmata with the MSD tags corresponding to the word form. **Context** can *disambiguate* lemmatization and tagging as well as improve accuracy on ambiguous and unseen words (Bergmanis and Goldwater, 2018; Bergmanis and Goldwater, 2019) because semantic meaning and morphosyntactic features can be inferred from contextual information to some extent. The second subtask in CoNLL-SIGMORPHON 2019 shared task (McCarthy et al., 2019) is about this: *morphological tagging* and *lemmatization* in context. Specifically, the input is a sentence, and the output is expected to be the corresponding lemma and MSD tag for each word in the sentence. See Table 1 for an example.

2.1.2 Morphological segmentation

Morphological segmentation is “the problem of how to split each word up into appropriate functional subparts” (Goldsmith et al., 2017) (p91). Not every word is separable without fuzziness. There are four cases of separability as illustrated with the following examples where + is used to indicate morpheme boundaries:

[4]The figure is cited from Hulden’s presentation slides (2015).

Segmentation	Tagging	Lemmatization	English Translation
dolar	N;3SG;Pnon;Nominative	dolar	dollar
dola+r	V;Positive;Aorist;3sg	dola	he/she wraps
dol+ar	V;Positive;Aorist;3sg	dol	it fills
do+lar	N;3PL;Pnon;Nominative	do	Multiple C (musical note)

Table 2: Four different ways to segment of the Turkish word *dolar*. *"+"* marks morpheme boundaries. Each of *dolar*, *dola*, *dol*, and *do* is a valid root in Turkish and $\emptyset$, *-r*, *-ar*, *-lar* are valid suffixes. $\emptyset$ means empty string. The example is adopted from Yildiz et al. (2016).

1. Completely separable, e.g. $played \rightarrow play + ed$ [5].

2. Separable, but with word form changes, though no change in the stem meaning, e.g. $sitting$ can be segmented in three different ways: (1) $sitting \rightarrow sit + ting$ (2) $sitt + ing$ (3) $sit + t + ing$.

3. Separable, but there are multiple ways of segmentation, and different segmentations result in different stems and meanings, e.g. the Turkish word *dolar* can be separated in four different valid ways as shown in Table 2;

4. Not really separable, e.g. it's questionable to segment $swam$ like this: $swam \rightarrow swim + ed$, and we don't have a better alternative.

The ambiguity and separability of the word also depends on the language. Ruokolainen et al. (2016) point out that "[m]orphological segmentation can be most naturally applied to highly agglutinative languages" (p95) in which multiple morphemes are usually concatenated together with clear boundaries.

For the second case of separability, the segmentation to keep the surface variations in the morphemes is **surface segmentation**, and if the changes to the morphemes during word formation are restored, it is **canonical segmentation** (Cotterell et al., 2016c).

There are two different segmentation mechanisms: flat segmentation and hierarchical segmentation. This distinction is more related to the derivational than inflectional process. For example, for the word $untestably$, there is only one way of **flat segmentation**: $un + test + able + ly$, but there are two different ways of **hierarchical segmentation**: $[un[test[able[ly]]]]$ and $[[un[test[able]]]ly]$, which are two different parsing tree structures as illustrated in Figure 3 (Cotterell et al., 2016b). The semantic ambiguity in the flat segmentation of the word can be clarified in the hierarchical segmentation (Cotterell et al., 2016b; Steiner, 2019).

2.1.3 Morphological disambiguation

In cases where multiple segmentations or analyses with different semantic and functional units are equally justified if the word is treated in isolation, like the Turkish example of *dolar* in Table 2 and the English example of *runs* in Figure 2(a), alternative segmentations or analyses are usually provided. If the context information is available, it can be applied to select the segmentation or

[5]When the format $X \rightarrow Y$ is used, it means the input to the task is X, the (actual or expected) output is Y, and $\rightarrow$ can be thought of as the model which takes the input and produces the output.

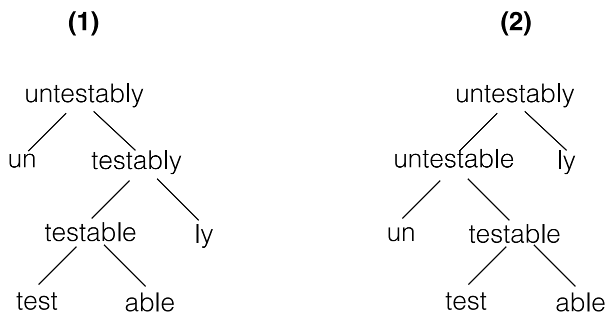


Figure 3: Parsing trees of two hierarchical segmentations of $untestably$. The trees were created by reference to Cotterell et al. (2016b)

lemma and MSD tag that work best for the context. This involves the task of *morphological disambiguation*: to pick out the segmentation or lemma and tag of a given word form that matches the context. For example, the following two example sentences containing *runs* are selected from *COCA*.[6] A morphological analyzer as the one in Figure 2(a) returns two possible analyses for the word form *runs*: (a) *run [V][PRES][3P][SG]*, (b) *run [N][PL]*, and a morphological disambiguator is expected to figure out that (a) matches the second sentence and it is (b) that is used in the first sentence.

(1) *Seeing those two home **runs** makes you question anybody who says baseball can't be exciting.*

(2) *He **runs** an investment company.*

Cotterell et al. (2018b) define a novel morphological disambiguation task, which they refer to as *disambiguation of syncretism in inflected lexicons*. Syncretism is the phenomenon where the same word form falls into multiple slots within one paradigm. For example, *run* is the non-3*rd*-person-singular form of the lemma *run* in present tense and the past participle form as well as the infinitive form of the lemma. Syncretism has been one of the cases neural models can't handle very well. What Cotterell et al. (2018b) did with syncretism disambiguation is to partition the count of different morphological analyses for the same given word form in the corpus. For example, suppose the word form *runs* appears in the corpus X times in total, Y out of the X times it is used as *run N;PL* and Z out of the X times it is used as *run V;PRES;3P;SG*, the task of synchratism disambiguation is to estimate the fractional count of the two different analyses of the word form *runs*, i.e. Y/X for *run N;PL* and Z/X for *run V;PRES;3P;SG*.

2.1.4 Learning of allomorphy and morphophonological variations

There are two tasks that have been intertwined with morphological segmentation at the phonology-morphology interface, i.e. *the learning of allomorphy and morphophonological variations*. The goal of these two task is to figure out how the various surface forms of morphological segments are related, including what are the underlying forms and what phonological changes are involved to derive the surface form from the underlying form (Cotterell et al., 2015). Table 3 provides the example of the regular plural suffixes for English nouns, which is realized as two orthographic

[6]https://www.english-corpora.org/coca/

Word	IPA transcription	Lemma	Word segmentation	IPA segmentation
cats	kæts	cat	cat+s	kæt+s
dogs	dɑgz	dog	dog+s	dɑg+z
fishes	fɪʃɨz	fish	fish+es	fɪʃ+ɨz

Table 3: Segmentation of English plural nouns *cats*, *dogs* and *fishes*

	INPUT			**OUTPUT**
Task	**source form**	**source tag**	**target tag**	**target form**
Inflection	run	-	V;3;SG;PRS	runs
Reinflection	running	(V;V.PTCP;PRS)	V;3;SG;PRS	runs
Lemmatization	running	(V;V.PTCP;PRS)	(V;NFIN)	run

Table 4: Examples of morphological inflection, reinflection, and lemmatization as a special case of reinflection. Tags in parenthesis may or may not be provided, and when they are provided, it is "labeled sequence transduction" (Zhou and Neubig, 2017b). "-" indicates the information is usually not specified.

allomorphs (i.e. *-s* and *-es* as shown in the column of Word segmentation) and three phonemic allomorphs (i.e. *-s*, *-z* and *-ɨz* as shown in the column of IPA segmentation). The task of figuring out what morphological segments belong to the same group, and often also picking out an underlying form for them, is *allomorphy extraction*.

The task of *morphophonological variation studies* is to handle and explain the variations in the surface forms. The example of English regular plural nouns involves voicing assimilation in the case of *cats* and *dogs*, and insertion of *ɨ* in the case of *fishes*. There are much fewer publications for this task, perhaps because computational processing of morphology usually deals with orthographic representation of words rather than phonemic representations.

2.2 Morphological generation

Morphological generation is a kind of string transduction or sequence-to-sequence transduction (Nicolai et al., 2015; Rastogi et al., 2016; Nicolai et al., 2018; Ribeiro et al., 2018; Makarov and Clematide, 2018b; Makarov and Clematide, 2018a; Wu et al., 2018). When labels are provided, the task is called *labeled sequence transduction* (Zhou and Neubig, 2017b).

2.2.1 Morphological (re)inflection

Morphological generation is the inverse of morphological analysis, as illustrated in Figure 2 on page 4. In Figure 2(b), the lemma form of the English verb *run* together with the MSD tag to be generated (`target tag`) is specified. The morphological generator is designed to produce the corresponding inflected form (`target form`). This task is *morphological inflection* (i.e. `lemma + target tag → target form`). If the given word form (`source form`) is not limited to the lemma, the task becomes *morphological reinflection* (Cotterell et al., 2017a). For the morphological reinflection task, the MSD tag of the given word form (`source tag`) may or may not be provided (i.e. `source form + source tag + target tag → target form`, or `source form + target tag → target form`). *Lemmatization* can be considered as a

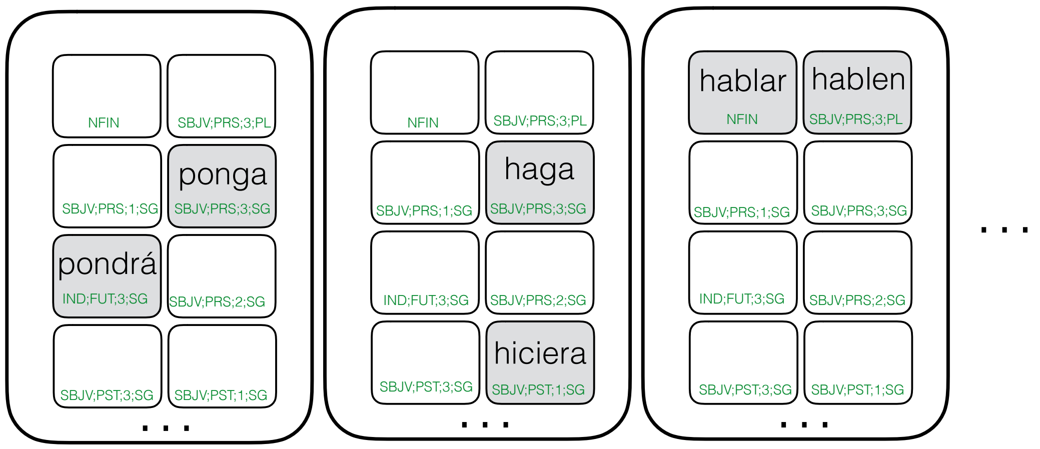


Figure 4: Illustration of the partial paradigm completion task. The examples are a fraction of Spanish verb paradigms. The PCFP task is to fill in all the missing slots in the given partially filled paradigms (Silfverberg and Hulden, 2018).

special case of *morphological reinflection*, where the form to be generated is limited to the lemma form (i.e. `source form + source tag → lemma`, or `source form → lemma`). Table 4 provides examples for the three tasks.

All the tasks can be **type-based**, i.e. to operate on the lexicon. In such tasks people have been working on, the MSD tags are usually explicitly specified. Annotated data at a relatively large scale is usually critical. In addition, semantic meaning and morphosyntactic features can be inferred from contextual information to some extent. **Inferring semantic and morphosyntactic information from context** makes it possible to train morphological inflection and reinflection models without relying on data explicitly annotated with MSD tags. For example, the second track of subtask 2 in CoNLL-SGIMORPHON 2018 shared task (Cotterell et al., 2018c) is to generate an inflected word form given the lemma of the word and the context it occurs in, as exemplified with sentence (3) where the expected form is *dogs* and the model is expected to generate this plural form of the noun:

(3) *The* _____ *(dog) are barking.*

2.2.2 Paradigm completion

Another morphological generation task is *paradigm completion*, also called *the partial paradigm cell filling problem* (PCFP) (Ackerman et al., 2009). In this task, incomplete inflection paradigms are provided, and the task is to fill in the missing cells in the partially filled inflection tables. Figure 4 illustrates the task where incomplete Spanish inflection tables are provided and the task is to fill in all the missing slots in all the paradigms. Such a task can be reduced to the (re)inflection task by using the given slots to predict each missing slot when there are more than one slot given in the paradigm. However, if each partial paradigm has only one given slot, the problem is more much challenging and requires a different approach.

2.3 Other tasks

In addition to the morphological analysis and generation tasks introduced above centering mainly on inflection, there are tasks involving **derivational morphology** as well (Cotterell et al., 2017d; Vylomova et al., 2017; Deutsch et al., 2018; Hofmann et al., 2020b; Hofmann et al., 2020a), though derivation is much less studied than inflection in computational morphology.

Historical text normalization (Bollmann and Søgaard, 2016; Bollmann et al., 2017; Korchagina, 2017; Robertson and Goldwater, 2018) aims to "convert historical word forms to their modern equivalents" (Robertson and Goldwater, 2018) (p720). This is also a task involving morphology. For example, the English word *said* may be spelled as *sayed*, *seyd*, *said*, *sayd*, etc., and the historical text normalization system is expected to normalize these different spellings to the modern spelling *said*, so that the historical text is more searchable and easier to process for downstream NLP tasks.

Deep learning models also provide new approaches to **simulate cognitive processing of morphology** and **quantify linguistic phenomena**. Kirov and Cotterell (2018) compare the learning curve of recurrent neural network models with cognitive linguistic findings of human language acquisition. Cotterell et al. (2018a) investigate and model under what conditions irregular inflections can survive in a language over the course of time. Williams et al. (2019) and McCarthy et al. (2020) attempt to quantify the grammatical gender system across languages.

2.4 Computational morphology: supervised, semi-supervised, unsupervised or reinforcement learning

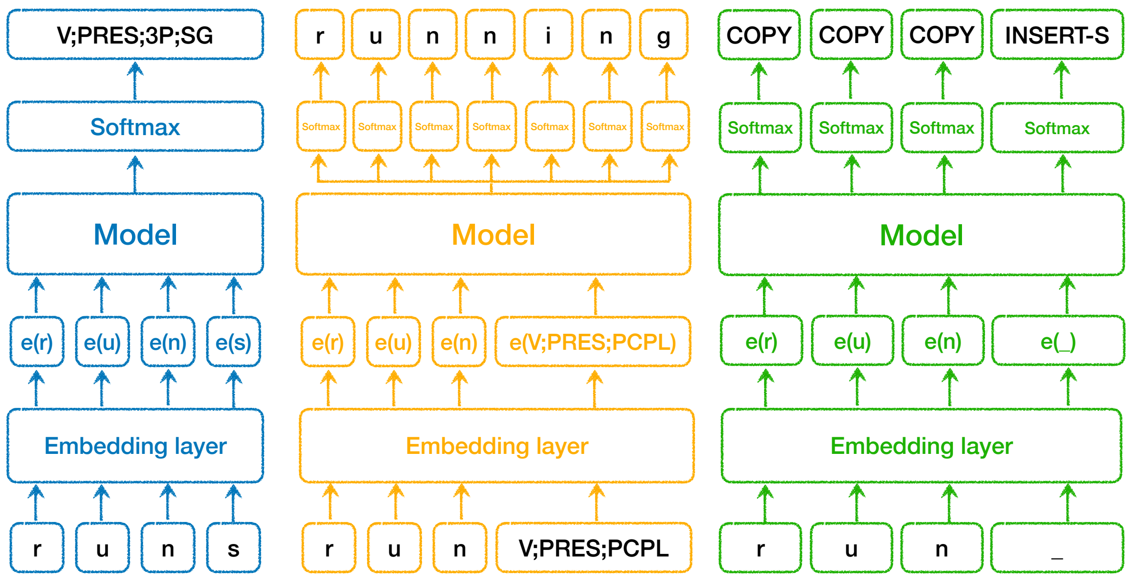
Another way to categorize tasks in machine learning, including computational morphology tasks, is by the use of annotated data. If a task relies on annotated data for training, it is a supervised task. If a task needs some annotated data, but can also use raw data or data not directly annotated for the task, it is semi-supervised. Unsupervised tasks operate on unannotated data like text crawled from Wikipedia. Reinforcement learning learns by taking actions in an environment to maximize cumulative reward. Currently there are few applications of reinforcement learning techniques to computational morphology.

3 Neural approaches for morphology processing

3.1 Three learning scenarios

The general setup of neural network models for computational morphology, is first to represent individual characters or other symbols depending on the desired level of granularity as one-hot vectors which are the inputs to the model, and then pass the one-hot vectors to an embedding layer. At the embedding layer, the model learns a weight matrix to convert each one-hot vector representation to a dense vector representation. The conversion from the input vector to the output vector is either through several steps or more simply in a single step. It's more common to have multiple steps and layers, and thus neural networks are usually called deep neural networks (DNNs). Based on how many output symbols are expected in relation to the number of input symbols, all the learning tasks can be generally interpreted into three different scenarios: many-to-one, many-to-many, and one-to-one scenarios (Bjerva, 2017). Different neural network architectures can be combined in different ways in each of the scenarios.

In the *many-to-one* scenario, the model first reads in the input text and gets the representation of the text. Then it makes the prediction, which is an output with only one symbol or one chunk



(a) Many-to-one **(b) Many-to-many** **(c) One-to-one**

Figure 5: Illustration of three different learning scenarios

of symbols, like POS or MSD tags as a chunk. The typical task for this scenario is **classification**. Morphological tagging is such a task if we treat each unique MSD tag as a class, as shown in Figure 5(a).

In the *many-to-many* scenario, the model first reads in the text, gets the representation of the text and then generates another sequence of symbols which may or may not be of the same length as the input. This scenario is usually referred to as **sequence-to-sequence transduction**. This scenario is illustrated in Figure 5(b), with the morphological inflection task. Other morphological generation tasks can also be naturally dealt with as this scenario.

In the *one-to-one* scenario, the model reads in the input text and outputs a prediction for each corresponding symbol in the input. The typical task for this scenario is **sequence labeling**. For example, in the morphological inflection task shown in Figure 5(c) where we need to generate the present participle for the English verb *run*, if we predict the inflected form by making the model to predict the edit actions to convert the input string to the expected output string, the correct operations should be *Copy*, *Copy*, *Copy*, and *Insert-s*.

3.2 Neural network architectures

The neural network architectures involved in the conversion process which happens in the model part in the plots in Figure 5, fall into four main types: feedforward, convolutional neural networks (CNNs), recurrent neural networks (RNNs) and the Transformer. The architectures can be mixed and combined with one another or with non-neural models like HMMs, CRFs, etc. to form more complicated models for different tasks. A softmax layer is usually applied to the output of neural encoding architectures for classification. For sequence transduction tasks, the encoder-decoder structure has been very successful. A lot of progress have been made to improve this structure recently. The last subsection will present the encoder-decoder structure.

3.2.1 Feedforward

Fully connected feedforward neural networks (Hornik et al., 1989), also called multi-layer perceptron, are non-linear and can be conveniently applied to classification problems as well as structured prediction problems. The feedforward neural network architecture has the drawback of requiring fixed length representations. One technique to represent unbounded number of symbols is through the bag of words method. The bag of words method simply adds up vector representations for multiple symbols (and perhaps also divides the sum by the number of symbols) to get the vector representation for the larger unit consisting of those symbols. However, the bag of words method has the problem of discarding order information in the input (Goldberg, 2016).

3.2.2 CNN

Convolutional neural networks (CNNs) (LeCun et al., 1995), also called convolution-and-pooling, are designed to pick out local predictors in a large structure and combine them to form a fixed representation of the structured input. Specifically, a non-linear function, called filter, is applied to each instantiation of a k-sized sliding window over the input to capture the important information within that window and transform it into a fixed-size representation. Then a pooling operation is conducted to combine the result from different windows into a single representation. Several convolutional layers can be applied in parallel. The most commonly used pooling techniques are max pooling and average pooling. This architecture can generate better vector representations than CBOW (Goldberg, 2016).

3.2.3 RNN

Recurrent neural networks (RNNs) are designed to capture the structured information in sequences of arbitrary length, which is especially suitable for natural language data processing. There are two main different types of RNNs: (1) vanilla RNN (Elman, 1990), (2) gated RNN which can be realized with either the Long-Short Term Memory (LSTM) architecture (Hochreiter and Schmidhuber, 1997) or the Gated Recurrent Unit (GRU) architecture (Cho et al., 2014b). Vanilla RNN represents the state at postion i as a linear combination of the input at this position and the previous state passed through a non-linear activation (usually ReLU or tanh). It is effective to capture sequential information, but is hard to train effectively due to the vanishing gradient problem as pointed out by Hochreiter et al. (2001). The LSTM architecture and the GRU architecture add "memory cells" controlled by gating components to tackle the problem in the vanilla RNN architecture.

Bidrectional RNNs (Schuster and Paliwal, 1997) encode the sequence forward and backward, and combine (usually by concatenation) the output of the forward pass and the backward pass at the same position in the sequence to get the representation for the position at hand, and thus capture both the past and future context (Graves, 2008; Goldberg, 2016). The bottom part of Figure 7 provides a sketch of the bidirectional RNN architecture.

3.2.4 Encoder-decoder structure and the Transformer

The encoder-decoder structure (Cho et al., 2014a; Sutskever et al., 2014), also called sequence-to-sequence architecture (or "seq2seq" for short), was initially designed for machine translation and then successfully applied to more generally sequence transduction problems. The encoder-decoder structure, illustrated in Figure 6, basically breaks down the input-to-output conversion process into encoder transformations and decoder transformations. The inner architecture of the encoder and decoder transformations can be formed from any or any combination of the three

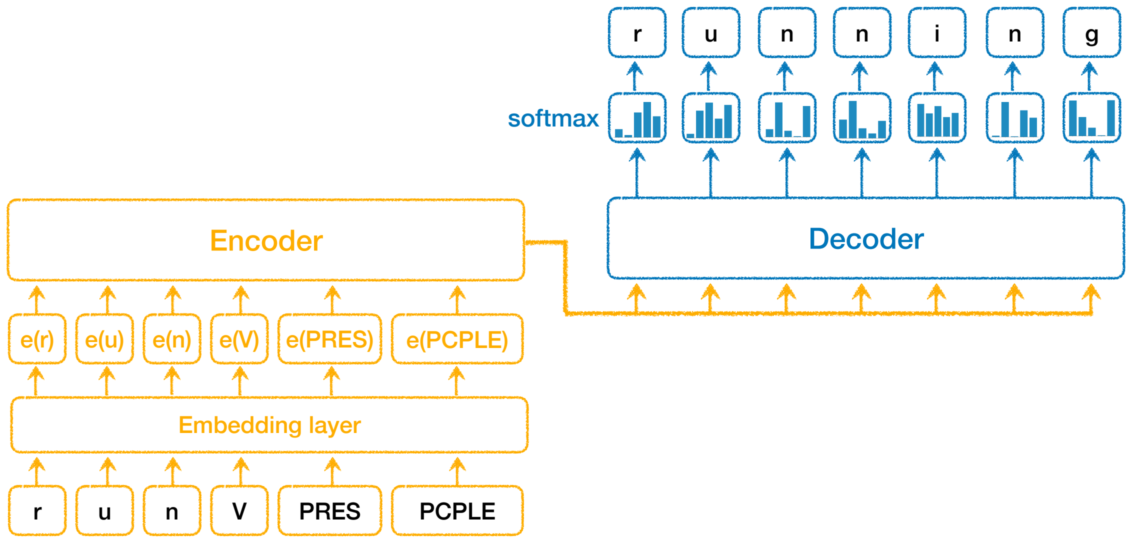


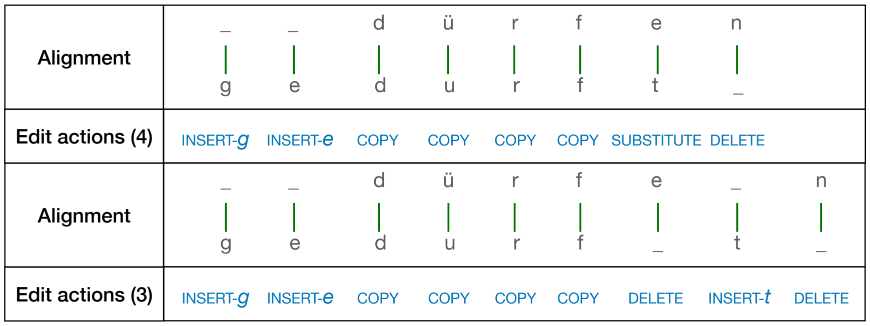
Figure 6: Illustration of the sequence-to-sequence architecture, with the morphological inflection task as an example: Given the English verb lemma *run*, generate its inflected form as the present participle.

basic structures introduced above. In a vanilla seq2seq architecture, the encoder converts the input sequence to a vector representation. The same vector representation is passed as input, usually together with other information based on the task, to the decoder at each time step to generate the output sequence. The vanilla architecture forces the encoded input vector representation to contain all the information from the input sequence for decoding and the decoder to be able to extract that information from the fixed-length representation from the encoder output. This structure works well in general, but the attention mechanism (Bahdanau et al., 2015) relieves the burden of the encoder and improves the model for sequence transduction.

The attention mechanism learns a weight vector applied to every encoding state for each decoding state. Figure 7 illustrates a seq2seq architecture with a bidirectional encoder, an attention mechanism and a decoder. The soft attention mechanism calculates the context vector as weighted sum of the encoder hidden states, while the hard attention mechanism selects certain encoder hidden states to use (Luong et al., 2015).

The RNN-based encoder-decoder architecture with attention has been very successful with string transduction. However, the sequential processing of the RNN is problematic: During the sequential processing, information may deteriorate when passing through the states over time. In addition, due to sequential processing, parallel computation is difficult to apply, making the architecture too slow for large scale use. The Transformer model (Vaswani et al., 2017) attempts to improve the seq2seq model to overcome its reliance on sequential processing. The Transformer still uses the encoder-decoder structure in its architecture, but it is stateless and does not use the RNN architecture. Instead, it uses multi-head attention to compute the input and output representations with self-attention layers as well as the combination of the input. It makes predictions with “encoder-decoder attention” layers which mimics the typical attention mechanism in previous seq2seq models.

4 Type-based computational morphology



Alignment	_ → g	_ → e	d → d	ü → u	r → r	f → f	e → t	n → _	
Edit actions (4)	INSERT-*g*	INSERT-*e*	COPY	COPY	COPY	COPY	SUBSTITUTE	DELETE	
Alignment	_ → g	_ → e	d → d	ü → u	r → r	f → f	e → _	_ → t	n → _
Edit actions (3)	INSERT-*g*	INSERT-*e*	COPY	COPY	COPY	COPY	DELETE	INSERT-*t*	DELETE

Figure 9: Alignment and edit action example for the German verb *dürfen* “to be allowed” and its past participle form *gedurft* “have/has/had been allowed”. Rows 1 and 2 show the alignment with four distinct edit actions. Rows 3 and 4 show the alignment with three distinct edit actions.

Type-based computational morphology refers to the computational processing of words out of context, like words in lexicon. The word form is provided in isolation, sometimes together with labels about morphosyntactic information.

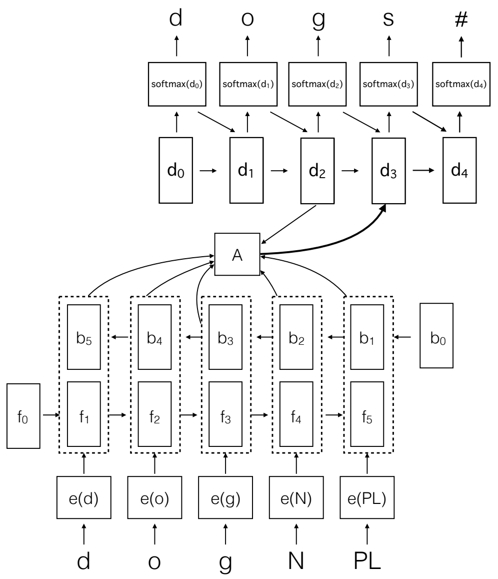


Figure 7: Illustration of the seq2seq architecture with an attention mechanism. The encoder illustrated here (bottom part) is bidirectional. The attention illustrated in the figure is that for generating the character *s*. The figure was created by reference to (Silfverberg et al., 2017).

4.1 Morphological generation with neural models – (re)inflection

Morphological generation, including inflection, reinflection, etc, belongs to the type of problem of mapping from one sequence to another. Before neural network models were applied, this type of problem has traditionally “been tackled by a combination of hand-crafted features, alignment models, segmentation heuristics, and language models, all of which are tuned separately.” (Yu et al., 2016)(p1307) Alignment and edit actions are still commonly leveraged in many neural network approaches (Aharoni et al., 2016; Aharoni and Goldberg, 2017; Makarov and Clematide, 2018c; Ribeiro et al., 2018; Makarov and Clematide, 2018b; Cotterell et al., 2017c; Kann and Schütze, 2017a; Chakrabarty et al., 2017; Kondratyuk and Straka, 2019). Edit actions usually include `COPY`, `DELETE`, `INSERT` and `SUBSTITUTE`. The `SUBSTITUTE` edit action can be treated as `DELETE` and `INSERT`. Figure 9 provides examples for the alignment with four and three distinct edit actions respectively.

Faruqui et al. (2016) is the first work to apply the **encoder-decoder** model to **inflection** generation. They use a bidirectional LSTM for the encoder, which takes the character sequence of the lemma form as input. The output of the encoder is fed into the LSTM decoder at each time step together with the decoder output from the previous time step. Their decoder does not use the attention mechanism. The authors experimented with learning separate models for each inflection type, or learning a shared encoder on the full inflection table and training a separate decoder for each inflection type. They also experimented with improving the model by using **unlabeled data**, i.e. to learn a language model over character sequences in a vocabulary and use the language model to rerank the decoding output by beam search or to interpolate the probability of the next word predicted by the language model with the probability of the next word predicted by the decoder. They report results obtained using **an ensemble of models** on the dataset created by Durrett and DeNero (2013) and extended by Nicolai et al. (2015), including full inflection tables for German, Finnish, Spanish, Dutch, and French in this dataset. Their model achieved results better than or comparable to previous state-of-the-art models on the same dataset (Durrett and DeNero, 2013; Ahlberg et al., 2014; Ahlberg et al., 2015; Nicolai et al., 2015). They also compare their model with the encoder-decoder model and the attention-based encoder-decoder model which doesn't incorporate the encoder output at each time step of encoding. They found that their model achieves the best performance. There are two main problems with Faruqui et al. (2016)'s models resulting from their design of training a separate model for each distinct MSD tag. First, they have to train as many models as there are MSD tags in the language, which will be cumbersome if the language has a lot of inflection types. Second, they need a lot of labeled data: no information can be shared between between different inflection types though similar rules may apply to the inflection for several MSD tags, and consequently, they need a lot of labeled data for each MSD tag in order to train good models. In addition, partially filled paradigms can't be utilized with their approach due to its reliance on complete inflection tables.

The shared task of SIGMORPHON in 2016 is about morphological (re)inflection, where morphological datasets of 10 languages with diverse typological characteristics were introduced. In **a labeled inflection task**, what's given is the lemma and a target tag, and the task is to predict the inflected form for the lemma corresponding to the target tag, i.e. *lemma + target MSD → target form*. In **a labeled reinflection task**, what's given is a source form, a source tag and a target tag, and the task is to predict the inflected form for the word corresponding to the target tag, i.e. *source form + source MSD + target MST → target form*. There is also **a more difficult version of the labeled reinflection task** which does not provide the MSD tag for the source form, i.e. *source form + target MSD → target form*. Nine teams submitted systems for this shared task. In general, the three teams who employed neural network models (Kann and Schütze, 2016a; Aharoni et al., 2016; Östling, 2016) outperformed by a large margin other teams who employed non-neural approaches of alignment and transduction pipelines or reduction of the problem to multi-class classifications with linguistic-inspired heuristics (Alegria

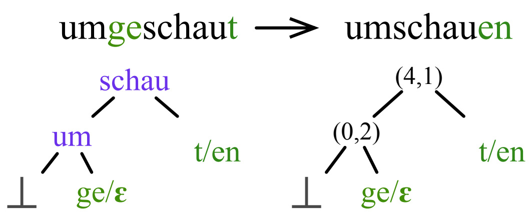


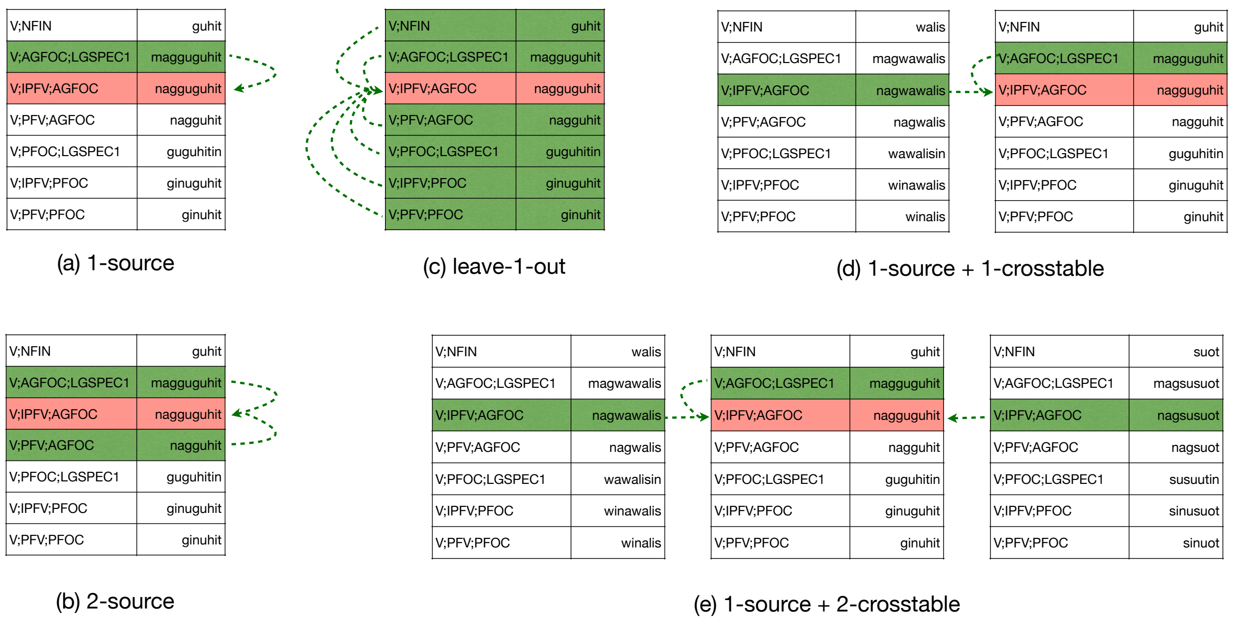
Figure 8: Edit tree to transform the German word *umgeschaut* "looked around" to its lemma *umschauen* "to look around". The blue nodes in the tree on the left are common substrings between the inflected form and its lemma form. These nodes are replaced with the span information in the edit tree on the right for generalization. *ge/ε* means deleting *ge*, *t/en* means substitute *t* with *en*, ⊥ means no edit action. This example is adopted from Müller et al. (2015)

and Etxeberria, 2016; Nicolai et al., 2016; Liu and Mao, 2016; King, 2016; Sorokin, 2016; Taji et al., 2016). The winning system (Kann and Schütze, 2016a) uses the same approach as Kann and Schütze (2016b), and applies the sequence-to-sequence architecture initially developed for machine translation successfully to the morphological (re)inflection task. Specifically, as illustrated in Figure 7, they use a GRU-based (Cho et al., 2014a) bidirectional encoder-decoder model (Sutskever et al., 2014) with a soft attention mechanism (Bahdanau et al., 2015), which takes as input the character sequence in the input word along with features in the source and target tags and predicts the target inflected form character by character. This formatting of the data is the key innovation. They used an ensemble of 5 models and selected the predicted form with the majority vote from the models as the final prediction. Different from Faruqui et al. (2016), they trained one model per language by treating each feature in the MSD tag as a separate symbol, and thus the approach has less strict requirement about the data. This system outperformed other models submitted for the same shared task by a large margin and pushed the morphological generation performance to a higher level across languages. Afterwards, the **bidirectional encoder-decoder with an attention mechanism** architecture has been widely applied to morphological generation tasks, and tweaks have been made to further improve the model. Kann and Schütze (2016b) propose the Perfect Observed Edit Trees (POET) method for correcting morphological reinflection predictions. An edit tree example is provided in Figure 8 on page 14. They evaluate combining this method with the neural model in Kann and Schütze (2016a). They find that adding the POET method can improve the neural model and outperform Faruqui et al. (2016)'s model by an even larger margin. However, the POET method relies on a large enough training dataset to cover the possible **edit trees** in the language.

The second-ranked neural model team (Aharoni et al., 2016) in SIGMORPHON 2016 shared task submitted two systems: One system is an **encoder-decoder** model with the encoder augmented with a bidirectional LSTM, and trains a network for each part-of-speech type. The model also has template-inspired components to align the training data first and train the network to either predict or copy a character at a given position. The other system from this team is based on neural discriminative string transduction, i.e. to first align the training data and get the sequence of **edit actions** to derive the inflected form from the source form, and then use the encoder-decoder model to predict the sequence of edit actions given the input sequence and the set of target MSD features, a hard-attention mechanism which has been tweaked more in later works like Aharoni and Goldberg (2017).

The third-ranked system (Östling, 2016) also uses a seq2seq model, but without an attention mechanism. They used multi-layers of LSTM for the decoder and train one model for each language to enable better parameter sharing. In addition to using the CNN architecture to extract features which are the inputs to the encoder-decoder layers of the LSTM architecture together with other feature vectors (e.g. character embedding, MSD vectors), Östling (2016) also experimented with using only the 1-dimensional residual network architecture (He et al., 2016) for encoding. They found that the accuracy is consistently improved by using convolutional layers together with LSTM for encoding and that purely convolutional encoding achieves even higher accuracy in many cases.

The vanilla encoder-decoder model has the limitation that decoding can only start after the whole input sequence has been encoded, and attention weights are treated as output of a deterministic function. In order to overcome this bottleneck, Yu et al. (2016) propose an online neural sequence-to-sequence model where the encoder is a unidirectional LSTM and beam search is used



V;NFIN	guhit
V;AGFOC;LGSPEC1	magguguhit
V;IPFV;AGFOC	nagguguhit
V;PFV;AGFOC	nagguhit
V;PFOC;LGSPEC1	guguhitin
V;IPFV;PFOC	ginuguhit
V;PFV;PFOC	ginuhit

(a) 1-source

V;NFIN	guhit
V;AGFOC;LGSPEC1	magguguhit
V;IPFV;AGFOC	nagguguhit
V;PFV;AGFOC	nagguhit
V;PFOC;LGSPEC1	guguhitin
V;IPFV;PFOC	ginuguhit
V;PFV;PFOC	ginuhit

(c) leave-1-out

V;NFIN	walis
V;AGFOC;LGSPEC1	magwawalis
V;IPFV;AGFOC	nagwawalis
V;PFV;AGFOC	nagwalis
V;PFOC;LGSPEC1	wawalisin
V;IPFV;PFOC	winawalis
V;PFV;PFOC	winalis

V;NFIN	guhit
V;AGFOC;LGSPEC1	magguguhit
V;IPFV;AGFOC	nagguguhit
V;PFV;AGFOC	nagguhit
V;PFOC;LGSPEC1	guguhitin
V;IPFV;PFOC	ginuguhit
V;PFV;PFOC	ginuhit

(d) 1-source + 1-crosstable

V;NFIN	guhit
V;AGFOC;LGSPEC1	magguguhit
V;IPFV;AGFOC	nagguguhit
V;PFV;AGFOC	nagguhit
V;PFOC;LGSPEC1	guguhitin
V;IPFV;PFOC	ginuguhit
V;PFV;PFOC	ginuhit

(b) 2-source

V;NFIN	walis
V;AGFOC;LGSPEC1	magwawalis
V;IPFV;AGFOC	nagwawalis
V;PFV;AGFOC	nagwalis
V;PFOC;LGSPEC1	wawalisin
V;IPFV;PFOC	winawalis
V;PFV;PFOC	winalis

V;NFIN	guhit
V;AGFOC;LGSPEC1	magguguhit
V;IPFV;AGFOC	nagguguhit
V;PFV;AGFOC	nagguhit
V;PFOC;LGSPEC1	guguhitin
V;IPFV;PFOC	ginuguhit
V;PFV;PFOC	ginuhit

V;NFIN	suot
V;AGFOC;LGSPEC1	magsusuot
V;IPFV;AGFOC	nagsusuot
V;PFV;AGFOC	nagsuot
V;PFOC;LGSPEC1	susuutin
V;IPFV;PFOC	sinusuot
V;PFV;PFOC	sinuot

(e) 1-source + 2-crosstable

Figure 10: Illustration of multi-source (re)inflection and cross-table (re)inflection. Green cells are the source. Red cells are the target. The figure is adopted from Liu and Hulden (2020a)

to marginalize a sequential latent variable for alignment in the decoding process. This allows the model to decide how much to encode before decoding each part. The model was evaluated on the tasks of abstractive sentence summarization and morphological inflection. The same dataset as Faruqui et al. (2016) was used for the morphological inflection experiments, and the model was trained separately for each type of inflection, the same setting as the factored model by Faruqui et al. (2016). It achieved comparable results as previous neural and non-neural state-of-the-art models (Faruqui et al., 2016; Durrett and DeNero, 2013; Nicolai et al., 2015) and the better performance on the Finnish data, whose stems and inflected forms are the longest, supports the advantage of this model. However, this model has not been applied to more tasks in computational morphology processing. One reason may be that the problem this model targets at, i.e. enabling the model to generate without waiting for having the entire input sequence encoded, is not a serious problem for morphology since most words are not very long. In addition, the Transformer model (Vaswani et al., 2017) overcomes the problem caused by very long sequences and produces the current state-of-the-art performance on the morphological inflection task (Vylomova et al., 2020).

Kann et al. (2017a) extend the typical **morphological (re)inflection** and use **multiple source** form-tag pairs to predict the inflected form for the target tag. Figure 10(a) illustrates the typical reinflection setup where a single source form is used to predict the target form. Figure 10(b) illustrates the case where two source form-tag pairs are used for predicting a target slot. Figure 10(c) illustrates the case where multiple source form-target pairs are used. Kann et al. (2017a)'s model is still the bidirectional encoder-decoder model with an attention mechanism, but uses one bidirectional encoder to encode each pair of source form and source tag for a lemma, and the decoder takes as input the attention-weighted sum of the encoder states. They use **full inflection tables**, and their experiments show that such a multi-source approach outperforms the typical single-source approach, especially when the multiple encoders share parameters. They also experimented with using multiple source form-tag pairs by concatenating all sources and encoding the concatenated form-tag pairs by one encoder. Their results indicate that the two ways of using multiple source form-tag pairs achieve comparable results.

The Transformer model was applied to the morphological (re)inflection task by Wu et al. (2020). In SIGMORPHON 2020 shared task 0 on morphological inflection, the Transformer model is the model architecture which produces the best performance (Vylomova et al., 2020; Liu and Hulden, 2020b). Moeller et al. (2020) compared the performance of the Transformer model (Vaswani et al., 2017) and the seq2seq model with exact hard monotonic attention (Wu and Cotterell, 2019) for inflection of noisy data in low-resource scenario. Their results support the advantage of the Transformer model as well.

4.1.1 Low-resource scenario

Though the performance of the seq2seq model is very impressive on the morphological (re)inflection task when there are abundant labeled data, the performance in the low-resource scenario drops dramatically. In order to improve the performance of neural network models in low-resource scenarios, efforts have be made to customize the model architecture for morphology tasks or to augment the training data by leveraging unlabeled data, data hallucination, labeled data from other languages etc.

Both machine translation and morphological (re)inflection are essentially string-to-string transduction tasks. The difference is that in morphological (re)inflection, a large part of the output is usually a copy of the input. Inspired by this difference, Aharoni and Goldberg (2017) present an encoder-decoder model with a hard-attention mechanism, which further improves the second system they created for SIGMORPHON 2016 shared task (Aharoni et al., 2016). The encoder is a bi-directional RNN which takes as input the concatenation of a forward RNN and a backward RNN states over the word's characters, and at decoding time, a control mechanism attends to a single input state and decides whether to write a symbol to the output sequence or advance the attention pointer to the next state from the bi-directionally encoded sequence. This model produces state-of-the-art results compared to previous neural and non-neural approaches. Its advantage is especially outstanding in the low-resource scenario. In addition, it's computationally more efficient than the soft attention mechanism for decoding: the model with hard monotonic attention allows linear decoding time while the decoding time for the soft attention mechanism is quadratic with regard to sequence length. This model has been widely adopted for low-resource languages, such as Makarov et al. (2017) who develop the winning systems with this architecture for CoNLL-SIGMORPHON 2017 shared task on morphological (re)inflection, Cotterell et al. (2017c) who apply the model to generate inflected word forms for principal parts morphological paradigm completion, and Junczys-Dowmunt and Grundkiewicz (2017) who apply the model to machine translation and automatic post-editing. However, this model is not truly end-to-end because it has a pipeline part: it relies on an external aligner to align the input string with the output string before training.

Makarov and Clematide (2018b) explain the hard attention mechanism as a neural transition-based system over edit actions and highlight the similarity of this approach with traditional weighted finite-state transducer (WFST) approaches. In addition, they replace the maximum likelihood estimation (MLE) training in Aharoni and Goldberg (2017) with **minimum risk training** (Och, 2003; Smith and Eisner, 2006; Shen et al., 2016), making the training process truly end to end. The model was evaluated on morphological inflection, reinflection and lemmatization tasks, and generated results comparable to or better than the state of the art. This is also the architecture used in the winning system for the first subtask in CoNLL-SIGMORPHON 2018 shared task. Makarov and Clematide (2018a) apply **imitation learning** to train neural transition-based approach and achieved similar effects.

Wu et al. (2018) use the hard non-monotonic attention with the seq2seq model. Their evaluation of the model on CoNLL-SIGMORPHON 2017 shared task 1 shows that this model achieves better results than the encoder-decoder with soft attention model in the high training data setting. Wu and Cotterell (2019) generalize the architecture of Wu et al. (2018) and propose **the exact hard monotonic attention mechanism for string transduction at the character level**, which enforces strict monotonicity, i.e. to attend to exactly one vector at memory at each output time step, and learns a latent alignment and transduction jointly. This seq2seq model with exact hard monotonic attention achieved state-of-the-art performance for CoNLL-SIGMORPHON 2017 shared task 1 at the high training data setting.

Ribeiro et al. (2018) thoroughly apply the idea of converting the morphological generation problem to a sequence labeling task by predicting edit operations, and present a technique for this transformation. They demonstrate that this approach performs comparable to the state of the art (Kann and Schütze, 2016a; Aharoni and Goldberg, 2017) for morphological inflection.

Zhou and Neubig (2017b) propose the **multi-space encoder-decoder model** for labeled sequence transduction. Compared to the common encoder-decoder architecture with an attention mechanism, their model features an intermediate step, the multi-space variational auto-encoder, which is a generative model that can process both continuous and discrete latent variables and enables the model to take advantage of both **labeled and unlabeled data** effectively. It was evaluated on the task of morphological reinflection, specifically the data for the 3rd subtask (i.e. $sourceform + targetMSD \rightarrow targetform$) of SIGMORPHON 2016 shared task, and achieved much better performance on most languages than the MED system (Kann and Schütze, 2016a) without model ensembling. When unlabeled data is used to train the model in a **semi-supervised** way through the multi-space variational auto-encoder, the result surpasses the MED system without model ensembling for all languages except Spanish. Kann and Schütze (2017b) is another work that found **unlabeled** data helpful. They take the **multi-task learning** approach by making the model to jointly predict the target inflected word form and map the unlabeled data with a special tag to itself, a task which they named as autoencoding. The use of unlabeled data improved the accuracy for the 8 languages they experimented with.

CoNLL-SIGMORPHON 2017 shared task 1 (Cotterell et al., 2017b) is on universal morphological inflection for 52 languages, which features 3 data-size settings for most languages: low (100 lemma-target MSD-target form triples for training), medium (1,000 lemma-target MSD-target form triples for training), and high (10,000 lemma-target MSD-target form triples for training) settings. There were 24 systems submitted for this subtask, all but one of which have a neural component, specifically the LSTM or GRU encoder-decoder architecture with an attention mechanism and all of which focus on the **low-resource** problem by model improvements like incorporating the model with the appropriate inductive bias, or **data augmentation** techniques like making use of unlabeled data or dummy data. The purpose of data augmentation is to increase the amount of training data and help the model learn the appropriate copy bias. The copy bias is critical in morphological (re)inflection because different from other sequence transduction tasks like machine translation, the transduction from the input form to the target inflected form usually involves copying the input symbol at many places. Bergmanis et al. (2017)'s system is the overall winning

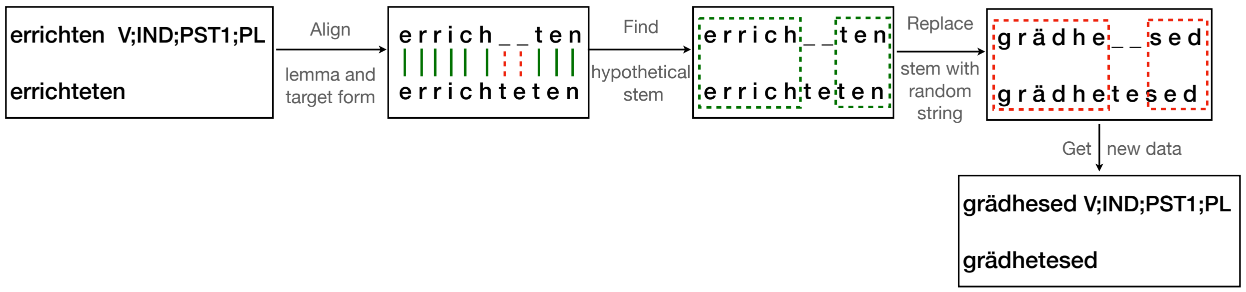


Figure 11: Illustration of the data hallucination method by Anastasopoulos and Neubig (2019). The only difference in Silfverberg et al. (2017)'s method is Silfverberg et al. (2017) take the longest common substring as the hypothetical stem, i.e. *errich* in this example.

system for the first subtask at high resource setting. In addition, they experimented with autoencoding of unlabeled data or random strings to make the model bias towards copying, as well as using labeled data in other languages to help training models for the current language (i.e. **cross-lingual transfer**). They found that autoencoding of random strings can improve the performance almost as well as autoencoding unlabeled words in both high and medium settings. What they do with autoencoding is to create additional training pairs by making an input string with a special tag like *COPY* to predict the input string itself. Zhou and Neubig (2017a) use **unlabeled data** with a variational autoencoder and latent variables (Zhou and Neubig, 2017b). Nicolai et al. (2017) try to augment the data by training a tagger with the training data and generate more labeled data by tagging more unlabeled data. Silfverberg et al. (2017) and Kann and Schütze (2017a) augment the data by creating artificial training data. Most systems use the soft attention mechanism, but the winning team (Makarov et al., 2017) develop systems based on the **hard monotonic attention mechanism** (Aharoni et al., 2016; Aharoni and Goldberg, 2017). Specifically, their first system extends the hard monotonic attention model with a copy mechanism and their second system is a neural state-transition model which learns to copy explicitly. Nicolai et al. (2017) use hard monotonic alignment by combining a discriminative string transduction model with the neural approach, though the system did not perform very well. The results of different teams indicate that the encoder-decoder architecture with an attention mechanism can handle morphological inflection well even when the training data is limited and that different model biases and data augmentation techniques compliment each other.

The data hallucination method by Silfverberg et al. (2017) is very effective for augmenting the training data in the low-resource scenario. This method was improved by Anastasopoulos and Neubig (2019) to take into consideration discontinuous stems as well. To be specific, both methods create hallucinated data following the steps illustrated in Figure 11 — First align the lemma with the inflected form, then find the hypothetical stem part, next generate a random string by uniformly sampling from the alphabet as dummy stem and replace the hypothetical stem with the dummy stem to form dummy training triples. The new dummy triple are added to the training set for data augmentation. The difference between Anastasopoulos and Neubig (2019) and Silfverberg et al. (2017) is in how the hypothetical stem is defined: Silfverberg et al. (2017) take the longest common substring between the source form and the target form as the stem while Anastasopoulos and Neubig (2019) take all the common substrings between the source form and the target form as the stem and thus include the discontinuous stem as well.

Spanish			Portuguese		
comer	**comemos**	V;IND;PRS;1;PL	comer	**comemos**	V;IND;PRS;1;PL
comer	coméis	V;IND;PRS;2;PL	comer	comeis	V;IND;PRS;2;PL
comer	comen	V;IND;PRS;3;PL	comer	comem	V;IND;PRS;3;PL;
comer	**comeríamos**	V;COND;1;PL	comer	**comeríamos**	V;COND;1;PL
comer	comeríais	V;COND;2;PL	comer	comeríeis	V;COND;2;PL
...	...	...	...	...	...

Table 5: A fraction of UniMorph inflection tables for *comer*, a verb used in both Spanish and Portuguese, meaning "to eat". Identical inflected forms are highlight in bold.

The first subtask of CoNLL-SIGMORPHON 2018 shared task (Cotterell et al., 2018c) is the same as that for the previous year: given a lemma form and a target MSD tag, the system is expected to predict the target inflected form in low, medium, high data conditions, and the system should be language-agnostic. The number of languages increased to 103 from 52. 27 systems were submitted for this shared task. Nearly all systems have a neural component, the same as in 2017. Though based on highly-ranked systems from previous years' shared tasks, further tweaks were made to the systems and 41 languages in the 52 included in 2017 shared task got better results in the low-resource setting. In order to deal with the **low-resource setting**, systems this year still try to learn **sequences of edit operations** or guide the neural model toward **copy bias** by creating artificial training data. Ensembling is also used by most systems to improve performance. The winning system (Makarov and Clematide, 2018c) uses the fully end-to-end transition-based model by Makarov and Clematide (2018b). Rama and Çöltekin (2018) develop a multilingual multi-task seq2seq model, but find that the model didn't surpass the baseline in high and medium data settings.

Character-level **language modeling** has been used to augment neural models for morphological (re)inflection. For example, Nicolai et al. (2018) leverage target language models derived from unannotated target corpora and make further improvements over the winning system for CoNLL-SIGMORPHON 2017 shared task 1 at low data setting (i.e. with 100 *lemma-tag-form* triples for training). Sorokin (2018) also attempt to improve the neural inflection model with character-level language modeling, but they only see marginal improvements.

Inspired by the analogy mechanism for inflecting unknown words by reference to known words, Liu and Hulden (2020a) propose to add inflected forms for the target slot in other inflection tables where this slot is given to the input source forms to predict the target form. This data manipulation method improves the performance of the Transformer model in extremely low-resource situations. Figures 10 (d) and (e) illustrate the cross-table data manipulation method. Though the interparadigmatic and intraparadigmatic aspects of analogy has been leveraged before neural network approaches have been applied to morphology (Ahlberg et al., 2014; Hulden, 2014; Liu and Hulden, 2017; Silfverberg et al., 2018a), Liu and Hulden (2020a) is the first attempt to explicitly provide both interparadigmatic and intraparadigmatic analogy sources for neural network models to learn from, and other existing work provide only the intraparadigmatic analogy source, leaving it for the model to infer the interparadigmatic analogical information implicitly.

4.1.2 Cross-lingual transfer

Languages share features, some related languages also share words and the inflection mechanisms are common across language groups. For example, Spanish and Portuguese are two languages that are closely related to each other. They shared words, features, and inflection mechanisms. Table 5 provides a fraction of UniMorph (Kirov et al., 2018) paradigms for *comer* "to eat" which is used in both Spanish and Portuguese, where we see much similarity. Could neural network models take advantage of the commonality across languages when multilingual resources are available?

Bergmanis et al. (2017) experimented with **cross-lingual transfer** in their system for CoNLL-SIGMORPHON 2017 shared task, but did not find multilingual resources more helpful than random strings. Rama and Çöltekin (2018) develop a multilingual multi-task seq2seq model for the first subtask of CoNLL-SIGMORPHON 2018 shared task, but the result of their system did not surpass the baseline in medium and high training data settings. The advantage of their system at the low-resource setting is not obvious either.

The first subtask of CoNLL-SIGMORPHON 2019 shared task (McCarthy et al., 2019) is about crosslingual transfer for inflection generation. Five teams submitted systems for this subtask. Three teams (Çöltekin, 2019; Hauer et al., 2019; Madsack and Weißgraeber, 2019) did not find obvious improvements in their systems by adding in cross-lingual data. Anastasopoulos and Neubig (2019) submitted the overall winning system output with the highest average accuracy and third-ranked average Levenshtein distance. Their system encodes the lemma and MSD tags separately: a bidirectional RNN is used to encode the lemma and a self-attention encoder (Vaswani et al., 2017) is used to encode the tags, after which two attention weight matrices are used to transform the representations into two sequences of context vectors, which are used by the recurrent decoder to generate the output in a two-step process. The two-step attention architecture for the decoder is novel and outperforms the baseline slightly. The training process of Anastasopoulos and Neubig (2019)'s morphological inflection method consists of 3 stages: the first stage trains the model to copy, i.e. be monotonic; the second stage is the main training phase when the model is train with both low- and high- resource language data as well as the dummy data created by the data hallucination method; and the last stage fine-tunes the model. They found that the relatedness between languages is closely correlated with the degree of transfer, and the same writing system is also critical for the cross-lingual transfer. These findings are similar to the findings in Kann et al. (2017b) and Jin and Kann (2017) who found that the degree of cross-lingual transfer for morphological inflection is strongly influenced by the relatedness between languages though Arabic data can also be leveraged to improve Spanish models. Anastasopoulos and Neubig (2019) also found that using data from multiple related languages can contribute even more to the low-resource language model. Peters and Martins (2019) submitted the overall second and third best systems in accuracy, which perform the best as to Levenshtein distance. Their models use sparse sequence-to-sequence modeling and attend to lemma and inflection tags respectively with **a two-headed attention mechanism** (Ács, 2018). In addition, their model is more interpretable. However, I think the better performance of their systems over the baseline indicates the advantage of their model, but does not provide enough support for crosslingual transfer effect in morphological inflection and can not exclude the possibility that the cross-lingual data only helped the model get a better copy bias.

4.2 Morphological generation with neural models – paradigm cell filling

Closely related to the morphological (re)inflection task is the partial paradigm cell filling problem (see Figure 4 on page 8 for illustration), as explained in section 2.2.2. When there are more than one slot available in partial paradigms, this task can be easily converted to the morphological (re)inflection task by taking given slots as the source and the missing slots as the target, following the data manipulation methods shown in Figure 10 on page 16, and all the techniques for improving morphological (re)inflection can then be applied. However, when there is only one slot given in each partially filled paradigm, the task is much more challenging (Malouf, 2016; Malouf, 2017; Silfverberg and Hulden, 2018).

Malouf (2016) and Malouf (2017) apply the neural network approach to the paradigm cell filling task. They utilize neural models but their model does not have the encoder-decoder architecture. What they do is to apply the idea of paradigm function, which maps a **lexeme** and an MSD tag to a target word form (Stump, 2001), and model the paradigm cell filling problem with an **LSTM generator**. At each time step the model takes as input a lexeme, an MSD tag set, and a partial predicted word form, where the lexeme and each MSD tag are represented as a two-hot vector which gets concatenated with the one-hot vector representation of the current character after dimension reduction, and output a probability distribution over the next character in the target word form. The probability of the target word form is the product of the probabilities for each predicted character for that target word form. They experimented on seven languages of different morphological complexity, having paradigms with 10% or 40% random slots missing and reported results on 10-fold validation. They found that their model can tackle the paradigm cell filling problem very well, and tried to interpret their results with linguistic reason and implications. The advantage of Malouf (2016) and Malouf (2017)'s model is that it represents the lexeme, which can be considered as the abstract underlying representation of the group of words forming the paradigm, and the target MSD tag as a two-hot vector, and thus **does not need to make use of the surface form**. However, one problem with their model is that the system can't generalize to words from unseen paradigms, i.e. it can only complete paradigms which it has seen examples in the training data.

Silfverberg and Hulden (2018) is in the same vein as Malouf (2016) and Malouf (2017) with more language acquisition consideration, but they use an **encoder-decoder model with soft attention** to solve the problem. More specifically, a 1-layer bidirectional LSTM is used to encode the given word form character by character, the source MSD features, and the target MSD features, and a 1-layer unidirectional LSTM with an attention mechanism is used to predict the target word form from the output of the encoder. Their model is more powerful for generalization, and can be used to complete paradigms which don't appear in the training data at all. In addition, Silfverberg and Hulden (2018) conduct their experiments in a more restricted setting: they evaluated the model with one, two or three slots given in the paradigm respectively, and they also evaluated a situation resembling a language acquisition setting more closely, i.e. when only the most frequent word forms in each paradigm are given. For the cases where more than one form are given in a slot, they generate training pairs by pairing up given slots and thus reduce the paradigm cell filling problem to a string transduction problem; for the case where only one form is given in the paradigm, they first train an **LSTM language model** at the character level conditioned on MSD features with the training data, and then use the language model to guide a mechanism of character dropout in the inflection model to produce a kind of pseudo-stem. In this way, they can get 2 forms per table and convert the task to a (re)inflection task. When trained on the same amount of training data, the encoder-decoder model by Silfverberg and Hulden (2018) achieved higher accuracy than their baseline model, i.e. the generator model by Malouf (2017), for most languages. For German nouns and Latin nouns in the case where only one form is given in each paradigm, the encoder-decoder model is slightly worse, which may be because the high degree of syncretism in the nominal declension of the two languages misleads the encoder-decoder model to copy too much, a problem that a model not relying on the surface word form, like the generator model, does not have.

Inspired by principal parts morphology, which states that a subset of forms in a paradigm, i.e. the principal parts, rather than a single form, provides enough information to determine the remaining forms of the paradigm (Finkel and Stump, 2007), Cotterell et al. (2017c) induce topology and try to decode the entire paradigm jointly. Specifically, they construct a **paradigm tree** by using Edmond's algorithm (Edmonds, 1967) to find the minimal directed spanning tree and the weight of the edges in the tree is the number of distinct edit paths between the characters in the pair of forms, which is calculated with the edit script procedure similar to Chrupala et al. (2008). They use the **encode-decoder LSTM model with hard monotonic attention** (Aharoni and Goldberg, 2017) to generate candidate target inflected forms and train the **graphical model** jointly to find the candidates for the missing slots that can best complete the paradigm. Their model was trained on more than 600 **complete paradigms**, an amount of data not available for most languages.

4.2.1 Low-resource scenario

The second subtask of CoNLL-SIGMORPHON 2017 shared task (Cotterell et al., 2017a) is about paradigm completion, though only two teams submitted systems and results for this subtask (Kann and Schütze, 2017a; Silfverberg et al., 2017). For training data, complete paradigms are provided, and for test, partial paradigms are provided and participants are expected to fill in the missing forms. High, medium, and low data settings, characteristic of CoNLL-SIGMORPHON 2017 shared task, are designed for this subtask as well. For the high data setting, most languages have around 200 full paradigms as training, and the number of inflected word examples varies as the size of paradigms are different for different languages. One of the participation teams (Kann and Schütze, 2017a) focused on solving this problem. They use the encoder-decoder model architecture, specifically a bidirectional GRU as the encoder and a unidirectional GRU with an attention mechanism for decoding. They address the problem essentially as a multi-source string transduction task, similar to Kann et al. (2017a) and Cotterell et al. (2017c). To improve the performance of the model, especially in **low-resource settings**, they use preprocessing and data augmentation. They preprocess the data by determining whether the language is predominantly prefixing or suffixing, using a special symbol to replace out-of-vocabulary (OOV) symbols at test time and copy the input symbol back for final prediction, a mechanism Silfverberg et al. (2017) also use. For **data augmentation**, they create training pairs by pairing up given forms in a paradigm with each other rather than using only pairs with the lemma and another inflected form, and add mappings of words to themselves and indicate it with a special tag, a technique they name as "autoencoding". They also experimented with first identifying the prefix or suffix of a word and replacing the stem part with random strings, a technique similar to the data hallucination method developed by Silfverberg et al. (2017). In addition, they made use of the principal parts idea that members of a paradigm have different dependence on each other, and try to select the best source form when multiple forms are given according to **edit trees** (Chrupala et al., 2008). Specifically, for each paradigm, they pair up given slots and build edit trees for each pair of source form and target form. In this way, each target MSD tag would have a set of source MSD tags with a set of edit trees associated with it.

The smaller the set of edit trees a source MSD tag has, the more important the source MSD slot is consider to be for the target MSD tag. Another approach Kann and Schütze (2017a) propose for the multi-source setting is fine-tuning, i.e. train one model for each language at each setting on all complete training tables, fine-tune the model on given slots in the partial tables at test time, and use the fine-tuned model to predict missing forms for the partial table. Silfverberg et al. (2017) is the other team to participate in this subtask, but it is not their focus. They also use the encoder-decoder model with an attention mechanism, but their encoder is a bidirectional LSTM and decoder is a unidirectional LSTM. A special symbol for OOV symbols at test time and the data hallucination method of replacing the stem with random strings is also used. Rather than using a majority vote to choose the best prediction from the output of multiple models, they use **weighted voting** with weights tuned by Gibbs sampling.

Kann and Schütze (2018) is also about paradigm completion in the **low-resource setting**, where being low-resource is defined as only a small number of **full paradigms** are available for training, same as CoNLL-SIGMORPHON 2017 shared task 2. Partial paradigms are provided at test time. Their model is based on the **encoder-decoder model with an attention mechanism** (Kann and Schütze, 2016b), improved by two methods which can be combined or applied separately: paradigm transduction and source selection. They found that in extremely low-resource settings where only 10 paradigms are available for training, the best performance was achieved by combining the method for source selection and the baseline of CoNLL-SIGMORPHON 2017 shared task, a non-neural model which extracts affixes based on character alignment; for cases with slightly more training data, like when 50 or 200 paradigms are available for training, combining the encoder-decoder model with both paradigm transduction and source selection achieves the best performance. For the paradigm transduction method, they first train the model on the training data as usual, and at test time, in addition to pairing up given forms, they also generate pairs where the given form with a special tag is mapped to itself, the same approach as autoencoding in Kann and Schütze (2017b), and continue training the model with parameters initialized on the training data. For selecting the source forms, the same approach is used in Kann and Schütze (2017a) as explained above.

4.2.2 Cross-lingual transfer

The paradigm completion task in Kann et al. (2017b) is a little different from the task in the papers mentioned above. In Kann et al. (2017b), the model generates all forms of the paradigm for a given lemma. They investigated the **cross-lingual transfer effect** for this task, i.e. to use training data for high-resource languages to help training models for low-resource language, where high-resource and low-resource are defined by the number of word inflection samples available for training. Their model is an **encoder-decoder** model, which is trained by appending a language tag to the input string. To be specific, the encoder is a bidirectional GRU and takes as input the concatenation of character embeddings for the lemma, MSD feature embeddings, and the representation for the language tag; the decoder is a unidirectional GRU with an attention mechanism. It's worth pointing out that MSD tags are broken down into features whose embedding representations are fed into the encoder. Their experiments on 21 languages from 4 language families indicate that cross-lingual transferring effect happens between languages that are closely related like Spanish and Portuguese as well as between languages that are very different, even with different writing systems, like Arabic and Spanish, though the more related the two languages are, the better morphological knowledge can be transferred. They also found that training with data from multiple languages can be helpful, though the performance improvements may not be obvious. For a tag which has been seen only once or hasn't appeared in the training data at all, if the high-resource language is very closely related to the low-resource language, the labeled data for the high-resource language can still help improve the prediction of the low-resource language. Though they have an experiment to separate the effect of true transfer from other effects by using a random cipher, it's not a strong support for cross-lingual transfer effect because the ciphered data also improve the prediction.

Jin and Kann (2017) use the same model for the same task as Kann et al. (2017b) and investigate the question of how and what knowledge gets transferred across languages. They conducted experiments on a subset of the data used in Kann et al. (2017b) by focusing on the transferring effect of letters and tags. They found that knowledge of character sequences and MSDs can be transferred and the improvement by using training data from a high-resource language, even a not closely related language, is larger than a pure regularization effect. For both Kann et al. (2017b) and Jin and Kann (2017), the paradigm completion problem is treated the same as the task of **morphological inflection**.

4.3 Segmentation with neural models

Most segmentation work to be discussed in this part involves segmenting parts of not only inflection, but also derivation and composition, which is the most common case for morphological segmentation, and derivation and composition tend to have a priority over inflection in labeled data for such tasks. For example, in the dataset published by Cotterell et al. (2016c), *touching* is not segmented to *touch ing*, rather it is taken as consisting of one morpheme. In other words, *touching* is treated as an adjective, rather than the present participle form of *touch*. The only exception is Moeller and Hulden (2018)'s work on automatic glossing for language documentation, where they not only segment the affixes and the stem, but also label the segments with morphosyntactic descriptions and English glosses, a combination of segmentation and labeling.

Cotterell et al. (2016c) distinguish surface segmentation and canonical segmentation. For surface segmentation, the words are split into multiple segments, and the concatenation of the segments are exactly the original word. In contrast, the segments are converted to canonical forms in canonical segmentation. Some ambiguity about segmentation boundary in surface segmentation can be avoided in canonical segmentation. However, the concatenation of the segments from canonical segmentation may be different from the original word and the canonical form for each morpheme needs to be identified. Identifying the canonical form for each morpheme is a task involving allomorphy learning discussed in section 2.1.4 on page 6. Figure 12 provides two possible ways for doing surface segmentation for the English word *funniest* and the canonical segmentation of the word.

4.3.1 Surface morphological segmentation

Wang et al. (2016) is about **surface morphological segmentation**. They treat the task as a **sequence labeling** or structured classification problem, i.e. for each character in the word, to classify it as B (beginning of a multi-character morpheme), M (middle of a multi-character morpheme), E (end of a multi-character morpheme), or S (a single-character morpheme), a one-to-one learning scenario. They proposed three **window-based LSTM** architectures for this task with different objective functions: a simple window LSTM model that makes individual predictions for characters, a multi-window LSTM model which also captures the label transactions, and a bidirectional multi-window LSTM model which has a backward pass in addition to the forward pass added to the

Surface segmentation 1	funn i est
Surface segmentation 2	fun ni est
Canonical segmentation	fun y est

Figure 12: Two possible surface segmentations and the canonical segmentation of the English word *funniest*. This example was created with reference to the example in Cotterell et al. (2016c).

multi-window LSTM model. They conducted experiments on the Hebrew and Arabic data from the dataset by Snyder and Barzilay (2008), and designed settings of different training data sizes by first ordering the data based on the frequency of the word and using the 25%, 50%, 70% or 100% most frequent of the training data for experiments. They compared the performance of their models with regular LSTM models and non-neural models which require heavy linguistic feature engineering, and their model, especially 10-model ensembling of the bidirectional multi-window LSTM model outperforms the regular LSTM model by a large margin, and it also achieves comparable results to non-neural models, by inferring linguistic knowledge from data with the window LSTM neural models.

Kann et al. (2018) focus on the morphological segmentation task for polysynthetic languages with only a small amount of labeled data. They improve the character-based **seq2seq model with an attention mechanism** by leveraging **unlabeled data** or random strings. They utilize the additional data by applying a method for multi-task training which **jointly** maximize the log-likelihood of an auxiliary autoencoding task (Kann and Schütze, 2017b) and the segmentation task, or by a **data augmentation** method which "treat the amount of additional training examples as a hyperparameter ... [to] optimize on the development set separately for each language" (p51). They also investigate the **cross-lingual transfer** effect by training one model for all related languages. In addition, they publish the dataset for four minimal-resource polysynthetic languages spoken in Mexico which they used for experiments. In general, their neural models did not achieve significant improvements over the CRF baseline they compared to.

Another related work aiming specifically at **low-resource languages** is Moeller and Hulden (2018) who place the morphological segmentation and labeling task in the context of language documentation, where usually only a very limited amount of data are available. Similar to Wang et al. (2016), for the CRF models they experimented with, they treat the task as a sequence-labeling problem, though their labels are much more details because their Beginning-Inside-Outside (BIO) tags are specified by morphological tags (as for fine-grained name-entity recognition where the model not only has to predict the scope of the named entity but also to specify what type of named entity it is) and they did not use a separate symbol for single-character morphemes. For the segmentation and labeling pipeline model, they still treat the segmentation task as a sequence labeling problem by using a CRF to predict the BIO tags, and then use a multi-class SVM to classify the morphemes as to their tags. In addition, they experimented with using an attention-enhanced encoder-decoder model with LSTM architectures for recurrence, which reads in the input word character by character and output the labels. This neural model did not surpass the CRF model with their designed features. The results of Kann et al. (2018) and Moeller and Hulden (2018) indicate that for conditions where we have extremely limited amount of labeled data, non-neural models with linguistic feature engineering still have an advantage, a finding also supported by Wolf-Sonkin et al. (2018) for morphological inflection within context.

Cotterell and Schütze (2018) incorporate semantics into segmentation. Their model does not have an overall neural network architecture. Instead, it consists of three factors: a transduction factor, which is a weighted-finite-state machine that computes probabilistic edit distance by looking at additional input and output context; a segmentation factor, which is an unnormalized first-order semi-CRF; and a composition factor, for which they experimented with addition and RNN respectively to combine morpheme embeddings to approximate the vector of the entire word. They found that **jointly** modeling semantic coherence and segmentation improves the segmentation performance, and that RNN is better than addition for vector composition.

4.3.2 Canonical morphological segmentation

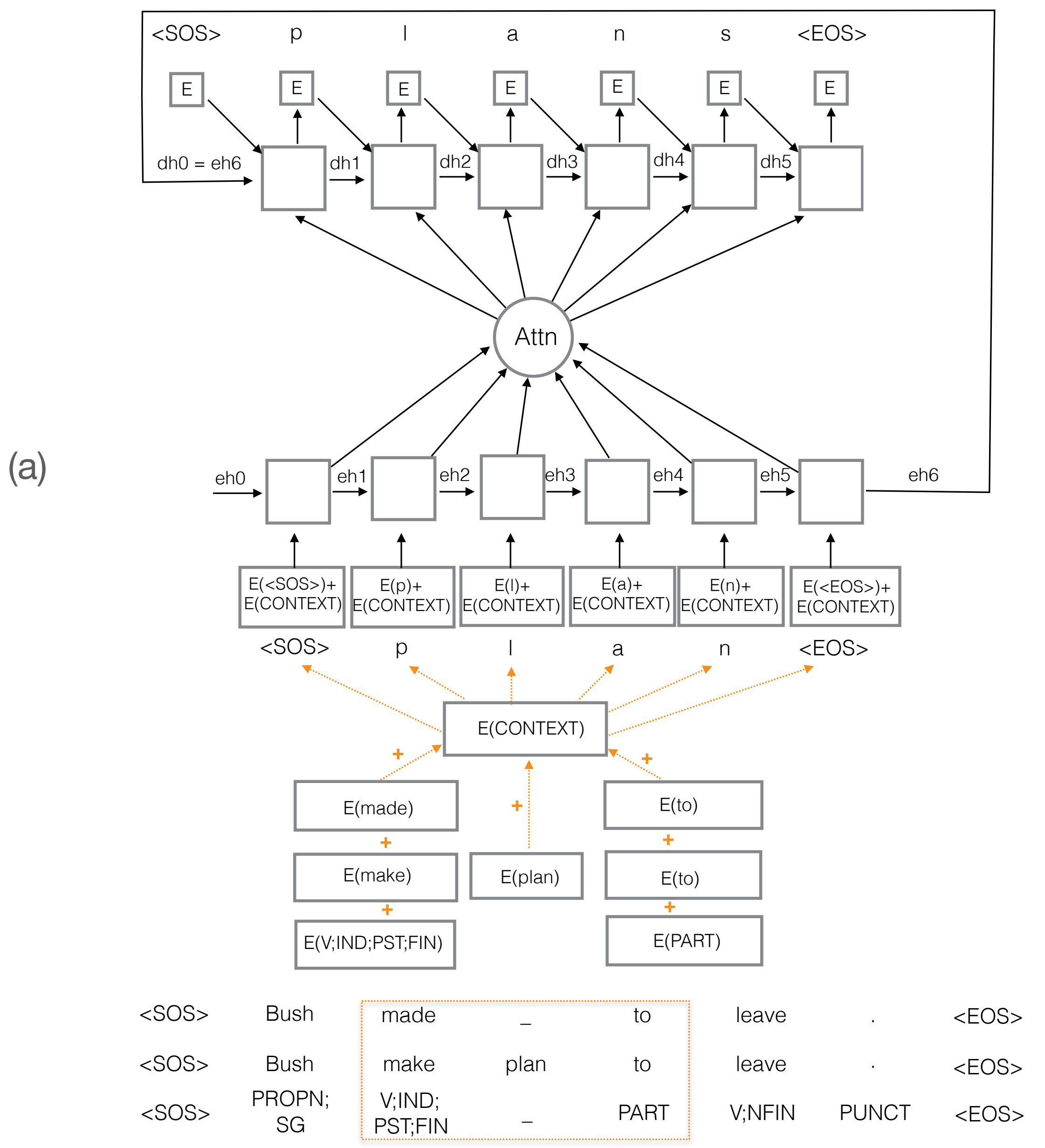
Cotterell et al. (2016c) introduce the canonical morphological segmentation task to convert a word to a sequence of segments in the “canonical” form. Their model is a joint model for transduction and segmentation with a finite-state transduction factor, a semi-Markov segmentation factor, and a sampling method to approximate the gradient for learning. An external dictionary is used in their model to check the validity of the segmentation. They evaluated their model on English, German and Indonesian and achieved state-of-the-art performance on the datasets they used and published for canonical morphological segmentation.

Kann et al. (2016) and Ruzsics and Samardzic (2017) are both about **canonical morphological segmentation** with **attention-enhanced encoder-decoder models**, where the morphological segmentation task is treated as a **string transduction** problem, the same as Moeller and Hulden (2018) did in their neural model experiment. Kann et al. (2016) add a neural reranker to the encoder-decoder model. The reranker uses external dictionaries and rescores the segmentation predictions with a feedforward network by incorporating explicit representations for entire segments with morpheme embeddings and thus making use of lexical information in addition to character information. They evaluated their model on the same dataset as Cotterell et al. (2016c), and achieved new state-of-the-art performance. Ruzsics and Samardzic (2017) don’t need the extra monolingual data employed by Cotterell et al. (2016c) and Kann et al. (2016) and improve Kann et al. (2016)’s result on the same data for Indonesian and German. They achieve this by integrating a **language model** over the canonical segmentation into the decoder. To be specific, they train the standard encoder-decoder model with soft attention and the language model over morphemes individually on the training data. The encoder-decoder model predictions and the morpheme language model scores are combined to find the best segmentation with a beam search algorithm.

The use of external dictionaries in Cotterell et al. (2016c) and Kann et al. (2016), and the use of language models in Ruzsics and Samardzic (2017) are efforts to utilize lexical meaning for morphological segmentation. All the work above did not make use of context information, though context is an important source of information for disambiguation when there are multiple ways to segment a word.

5 Token-based computational morphology

After the success of neural network approaches with type-based morphological processing, more and more work starts to explore the performance of neural models to extract morphosyntactic infor-



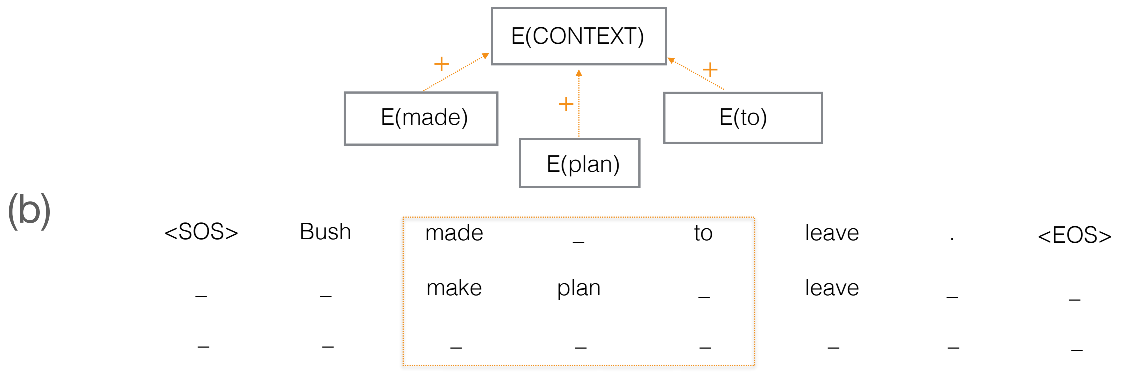


Figure 13: Illustration of the seq2seq model applied to inflection in context. (a) illustrates the whole model architectures with labeled context (i.e. Track 1) (b) illustrates the contextual embedding for unlabeled context (i.e. Track 2). The figures are adopted from Liu et al. (2018).

mation from sentential context. We see more applications of neural network approaches to token-based computational morphology in the past several years, especially morphological (re)inflection, lemmatization and morphological tagging in context.

5.1 Neural models and morphological (re)inflection in context

The second subtask of CoNLL-SIGMORPHON 2018 shared task (Cotterell et al., 2018c) is about morphological inflection in context, i.e. given a lemma and a sentential context it occurs in, including inflected forms of contextual words either with or without the corresponding lemma form and MSD tag, predict the correct inflected form for the lemma satisfying the context. The bottom three lines in Figures 13 (a) and (b) provide examples for the two versions of the task. This shared task also has three training data conditions: low, medium, and high. There were 6 submissions, all of which use neural models. The neural baseline system, illustrated in Figure 13, has an **encoder-decoder with a soft attention mechanism architecture**, where the prediction of the inflected word form is conditioned on sentential context, different from type-based morphological inflection whose decoding is conditioned on the explicitly specified MSD tags. The context embedding is defined as a concatenation of embeddings of the current lemma at both word level and character level, the left and right context word form at the word level. When the lemma forms and the MSD tags for the left and right context words are given, They are also concatenated into the context embedding. Most submissions are developed on top of this baseline system, and define the context either as the baseline or expand the context by including more words before and after. Kementchedjhieva et al. (2018)'s system achieved the best overall accuracy of 49.87%, and specifically it won this task in the high data setting for both tracks and the medium data setting for track 1. This system is an augmentation of the baseline system in three aspects: (1) a wider context – an LSTM is used to encode the entire available context, (2) **multi-task learning** with the auxiliary task of MSD prediction for track 1, i.e. when the MSD tags for the contextual words are given, (3) first training models in a **multilingual** fashion with random groupings of two to three languages, followed by monolingual fine-tuning for each specific language. An ensemble of the five best models is used to make the final prediction by majority vote. Makarov and Clematide (2018c) apply the system they propose in Makarov and Clematide (2018a), which is a **neural transition-based transducer with a designated copy action** trained in a fully end-to-end fashion with the **immitation learning** framework (Daumé et al., 2009; Ross et al., 2011; Chang et al., 2015). This system won the task in the medium data setting for track 2 and the low data setting for both tracks. It's worth pointing out that even the overall best performing system only achieved an average accuracy of lower than 50%, indicating that the problem is difficult and there is a large space for improvement.

5.1.1 Low-resource scenario

In order to utilize raw token-level data to improve morphological inflection in context in the **low-resource scenario**, Wolf-Sonkin et al. (2018) propose a **generative directed graphical model**, which is a full-fledged **language model** that can use both **labeled and unlabeled data**. The model consists of an LSTM language model for MSD prediction, a lemma generator defined as a conditional LSTM language model over characters, and a morphological inflector as a seq2seq model with a soft attention mechanism. When the model is trained with the raw text, unobserved lemmata and MSDs can be approximated by posterior inference over latent variables based on the wake-sleep algorithm (Hinton et al., 1995). They found that agglutinative languages tend to benefit more from the use of raw text. One of their baselines is a probabilistic finite-state transducer system

Input sentence	He	told	the	children	an	encouraging	story	.	
Lemmatization	he	tell	the	child	a	encouraging	story	.	
Input sentence	He	is	encouraging	the	children	with	a	story	.
Lemmatization	he	be	encourage	the	child	with	a	story	.

Table 6: Examples for lemmatization in context. The context is needed to determine whether *encouraging* should be lemmatized as *encourage* or not.

(Cotterell et al., 2017b). They found that the FST system is competitive with neural models when the training data is limited and that neural models can learn non-concatenative morphology better than the FST system. Though they claim to be the second attempt to perform semi-supervised learning for a **neural inflection generator**, preceded by Zhou and Neubig (2017b), it remains unclear why they didn't count other work trying to make use of unlabeled data as semi-supervised. Perhaps, they were talking about the model structure. Their model and Zhou and Neubig (2017b)'s model both highlights a special design in the model architecture for leveraging unlabeled data, which is missing in models used in other work on neural morphological inflection.

5.1.2 Joint learning

Vylomova et al. (2019) tackle CoNLL-SIGMORPHON 2018 shared task 2 with morphological tags given for contextual words. They propose **a structured neural model**, which explicitly reconstructs MSDs and then uses the predicted MSDs and lemmata to predict the corresponding inflected forms. Their tagger is a neural conditional random field and the morphological inflector is a neural encoder-decoder model with hard attention (Aharoni and Goldberg, 2017). They compared their system with a system that directly predicts the inflected forms without relying on any morphological annotation, as well as the shared task baseline system (Cotterell et al., 2018c) and the monolingual version of the shared task winning system (Kementchedjhieva et al., 2018). Their lemmatization results outperform other models for most languages. They found that when lemmatization is conditioned on the ground-truth MSD tags, the accuracy can be much higher. These results indicate that **jointly** predicting morphological tags is helpful for morphological inflection in context. This supports incorporating linguistically motivated information into neural models.

5.2 Neural models and context-sensitive lemmatization

Table 6 provides examples of context-sensitive lemmatization where the task is to convert the words in the given sentence to the corresponding lemma forms satisfying the context. Neural approaches to lemmatization in context have been treating the task either as **a classification problem of selecting and applying an edit tree that produces the lemma from the inflected form** or as **a string transduction problem of mapping the input character string of the inflected word forms to the lemma string**.

Chakrabarty et al. (2017) treat the task as an **edit tree classification** problem with two successive bidirectional RNNs. The first bidirectional RNN builds the syntactic embedding, i.e. to generate the composite vectors for all words in the training set, and the second bidirectional RNN learns the representation of the local context and is trained to predict the most likely lemma corresponding to the applicable edit trees. In other words, the second bidirectional RNN predicts the lemma for the word in context by identifying the correct edit tree as a classification problem. They experimented with both LSTM and GRU for the recurrent neural network architecture, and found that

run	run	V;NFIN	speak	speak	V;NFIN
run	ran	V;PST	speak	spoke	V;PST
run	running	V;V.PTCP;PRS	speak	speaking	V;V.PTCP;PRS
run	runs	V;3;SG;PRS	speak	speaks	V;3;SG;PRS
run	run	V;V.PTCP;PST	speak	spoken	V;V.PTCP;PST

Table 7: UniMorph inflection table examples with English verbs *run* and *speak*. For each verb, the first column is the lemma, the second column is the inflected form and the third column is the corresponding MSD tag.

they performed equally well. The system outperformed Lemming (Müller et al., 2015) and Morfette (Chrupala et al., 2008), two previous state-of-the-art models, both of which are non-neural models learning to do lemmatization and morphological tagging jointly, on 9 languages except Bengali. The annotated Bengali dataset is another contribution by Chakrabarty et al. (2017).

5.2.1 Low-resource scenario

The model *Lematus* (Bergmanis and Goldwater, 2018) is a lemmatizer based on the standard **encoder-decoder with soft attention mechanism** (Kann and Schütze, 2016a), and incorporates sentential context. The system was developed especially with the consideration to improve the lemmatization of **ambiguous words** and **unseen words**. They also conducted experiments with different amount of training data, and experimented with different ways to represent context, including word-level context representations that use all words in the left and the right of the sentence, character-level context representations that use 0, 5, 10, 15, 20, or 25 characters in the left and right of the sentence, and sub-word unit context representations by byte pair encoding. For the 20 languages from UD v2.0 they tested on, *Lematus* outperformed Lemming (Müller et al., 2015), Morfette (Chrupala et al., 2008), Chakrabarty et al. (2017) and a lookup-based baseline which predicts the most frequent lemma form for observed data and the word itself for unseen words, in full-data setting and comparable results in limited data setting (i.e. with 10k training examples). They found that including context can improve the performance of the system significantly in both full data setting and limited data setting, and that context can help more with the lemmatization of ambiguous words than unseen words. Their cross-linguistic analysis indicates that the proportion of unseen words in a language is anti-correlated to the proportion of ambiguous words in that language. Bergmanis and Goldwater (2019) attempt to improve the performance of *Lematus* when the training data is limited by using UniMorph (Kirov et al., 2018) inflection tables (see Table 7 for example) and Wikipedia text to create additional training data. Their results indicate that this **data augmentation** method is useful in low- and medium-resource settings.

5.2.2 Joint learning

Lemmatization and morphological tagging in context are mutually dependent and can provide information for each other to exclude unlikely choices in cases where there are multiple ways for lemmatization or tagging. It's very common to conduct lemmatization and morphological tagging together, which is usually referred to as morphological analysis where morphological analysis is used in the narrower sense. Table 8 adds the ground-truth MSD tags to each word in the examples in Table 6. The *encouraging* in the first example is an adjective whose lemma form should be *encouraging*, while the *encouraging* in the second example is a verb in the present participle

Input sentence	He	told	the	children	an	encouraging	story	.	
Lemmatization	he	tell	the	child	a	encouraging	story	.	
MSD tagging	PRON;NOM;SG	V;PST	DET;DEF	N;PL	DET;IND	ADJ	N;SG	PUNCT	
Input sentence	He	is	encouraging	the	children	with	a	story	.
Lemmatization	he	be	encourage	the	child	with	a	story	.
MSD tagging	PRON;NOM;SG	AUX;SG;3;PRS	V;PRS;PCTP	DET;DEF	N;PL	ADP	DET;IND	N;SG	PUNCT

Table 8: Examples for lemmatization and morphological tagging in context.

form whose lemma form should be *encourage*. Müller et al. (2015) provides empirical evidence that joint modeling of lemmatization and morphological tagging is mutually beneficial. There are also neural models developed for **joint** learning of lemmatization and morphological tagging, e.g. Kondratyuk et al. (2018), Malaviya et al. (2019), McCarthy et al. (2019), which provide further support for the effectiveness of learning context-sensitive lemmatization and morphological tagging jointly.

LemmaTag (Kondratyuk et al., 2018) is a bidirectional RNN architecture with character-level and word-level embeddings that jointly generates MSD tags and lemmata for sentences. Specifically, the model consists of three parts: (1) The encoder with the structure of Chakrabarty et al. (2017). A bidirectional RNN gets the character-level representation of each word (Heigold et al., 2017) and the embedding for each character are concatenated to form the character-level embedding for the word. The character-level embedding is summed up with the word-level embedding obtained from a word-level embedding layer to form the final word embedding for the word. The final word embeddings are the input into two layers of bidirectional RNN with residual connections to get the context representation. The encoder is shared by the tagger decoder and the lemmatizer decoder. (2) The tagger applies a fully-connected layer to the representations obtained with the encoder to generate the tags. (3) The lemmatizer applies an RNN decoder with soft attention to character encodings, summed embeddings for the current word, predicted tag features and context representations to generate the lemma form character by character. The final loss function is defined as the weighted sum of the losses of the tagger and the lemmatizer. This system achieved state-of-the-art performance on tagging and lemmatization in Czech, German and Arabic, and comparable performance to the state of the art for English. They point out that jointly learning lemmatization and morphological tagging by sharing encoder parameters and providing the predicted tags as input to the lemmatizer improves the performance of both lemmatization and tagging, especially for morphologically rich languages. However, in languages with less complex morphology like English and German, it may hurt the performance of the tagger to share the encoder parameters.

The joint neural model for lemmatization and morphological tagging developed by Malaviya et al. (2019) has a similar architecture with Vylomova et al. (2019), i.e. a **structured neural model** consisting of a morphological tagger part and a lemmatizer part. The morphological tagger is the SPECIFIC model from Cotterell and Heigold (2017) reimplemented by Malaviya et al. (2018). The lemmatizer is a seq2seq model with hard attention mechanism. Greedy decoding and crunching (May and Knight, 2006) were experimented with to get the tagging result used for lemmatization. The crunching scheme is a heuristic to approximate true joint learning by first predicting the k-best tags with the morphological tagger and then incorporating the predicted tags into the lemma generation process. The crunching schema achieved slightly higher average accuracy. They train the models with jackknifing (Agić and Schluter, 2017) to avoid exposure bias in order to improve the lemmatization result. The evaluation focuses on the lemmatization results. Their lemmatization results outperform *Lematus* (Bergmanis and Goldwater, 2018), UDPipe (Straka and Straková, 2017), Lemming (Müller et al., 2015) and Morfette (Chrupala et al., 2008). Similar to the finding of Kondratyuk et al. (2018), their error analysis indicates that jointly learning morphological tagging and lemmatization is especially helpful for languages with more complex morphology. They also found that when the training data is limited for lemmatization, joint learning of morphological tagging and lemmatization is helpful. Similar to Vylomova et al. (2019), Malaviya et al. (2019)'s experiments show that when the ground-truth MSD tags are given to the lemmatizer, the lemmatization accuracy can be much higher.

The second subtask of SIGMORPHON 2019 shared task (McCarthy et al., 2019) is on morphological tagging and lemmatization in context. The non-neural baseline for this task is Lemming (Müller et al., 2015) and the neural baseline is Malaviya et al. (2019). 16 systems were submitted for this subtask, all of which use neural network models. The most successful systems are implementations of variations of multi-headed attention, multi-level encoding and multiple decoder, and use BERT (Devlin et al., 2019) pretrained embeddings for contextual representations. Kondratyuk (2019)'s system achieved the highest average accuracy and F1 score for morphological tagging and the second average accuracy in lemmatization. It consists of a shared BERT encoder and predicts lemmata and MSD tags jointly. To be specific, BERT multilingual cased tokenizer is first applied to each token in the input sentence, which is then encoded with pretrained multilingual cased BERT base model. Two separate layer attention instances similar to ELMo (Peters et al., 2018) as defined in UDify (Kondratyuk and Straka, 2019), generate embeddings specific to each task. A bidirectional LSTM with residual connection is applied to the sequence of characters for each word to encode the morphological information, and two successive layers of word-level bidirectional residual LSTMs are leveraged to compute over the entire task layer attetion for each task. For lemmatization decoding, it is treated as a **sequence classification** task to find the **edit actions** to transform an inflected word to its lemma and a feedforward layer is applied to the lemmatizer LSTM final layer to get the result. For morphological tagging, a feedforward layer is applied to jointly predict the factored and unfactored MSDs. Factored MSDs treat the MSD tags at the feature tag level, while unfactored MSDs treat the whole chunk of a MSD tag as one tag. For example, for a MSD tag *N;PL*, there are two factored tags *N* and *PL* while it is one unfactored tag. The factored tags are used to improve training and the tagging prediction are the output of the unfactored tags. Their results support the **cross-lingual transfer** effect for boosting lemmatization and morphological tagging because they achieve better performance by fine-tuning multilingual BERT on all available treebanks for all languages over fine-tuning monolingually. In addition, they found that the additional character-level and word-level LSTM layers for embedding can improve the performance even further.

The winning system of the lemmatization task in SIGMORPHON 2019 shared task 2 is Straka et al. (2019), which ranks the second for morphological tagging performance. This system is an improvement of UDPipe 2.0 (Straka, 2018) which uses a single joint model for morphological tagging and lemmatization with bidirectional LSTM models. UDpipe also treats lemmatization as a **classification** problem by making the model to predict generation rules to convert the inflected word forms to their corresponding lemmata. The improvements are from 3 aspects. First, similar to Kondratyuk (2019), they use pretrained BERT embedding as additional inputs to the network. Secondly, also similar to Kondratyuk (2019), they regularize the model by predicting factored tags though they take the tagging prediction as the unfactored tags. Thirdly, rather than concatenating all languages like Kondratyuk (2019), they merge treebanks for the same language. The success of Kondratyuk (2019) and Straka et al. (2019) indicates the effectiveness of predicting **edit operations** rather than character by character for lemmatization, a finding also supported by Yildiz and Tantuğ (2019) who submitted the third ranked lemmatization system for this task.

Shadikhodjaev and Lee (2019) is another team participating in SIGMORPHON 2019 shared task 2. They developed a **transformer-based seq2seq model** for lemmatization and a bidirectional LSTM model with attention for morphological tagging. They produce the lemma for each word in the sentence, and then use the predicted lemma and the given word form together to predict the MSDs. This way of using lemmatization output to improve tagging as a pipeline is not common, and the more common way is the other around. Their system ranked fifth in lemmatization and sixth in tagging.

Another work to jointly learn lemmatization and morphological tagging is Akyürek et al. (2019). This system focuses on morphological tagging and the tagging results outperform Cotterell and Heigold (2017) and Malaviya et al. (2018). It's interesting to point out that they found that making the decoder to jointly predict the lemma form and the factored MSDs produces generally worse results than decoding only the MSDs, especially in the low-resource setting and for morphologically more complex languages. They also found that predicting factored MSDs is advantegous over predicting unfactored MSDs when the training data is limited and the language has rich inflections, and that using related high-resource language data to boost **low-resource language** training is helpful. The structure of their system is as follows: They use a unidirectional LSTM at the character level to get the word embedding for each word; a forward LSTM at the word level to read in words to the left of the current word and a word-level backward LSTM to read in words to the right of the current word, and take the concatenation of the forward and backward LSTMs as the context representations; another unidirectional LSTM to encode the predicted morphological tagging of the previous two words. For the decoder part, they use an LSTM to generate the lemma character by character followed by the MSD features. In their model, they did not make use of the lemma information other than in the decoding process, which explains why the model generally performs better when it only predicts the MSDs. Another contribution of the work is that they published a new Turkish morphology dataset with a high (96%) inter-annotator agreement.

Closely related to joint learning of lemmatization and morphological tagging is **morphological disambiguation**, as shown with the examples in Table 8 where we need to disambiguate *encouraging* in order to get the correct lemma form and MSD tags. Yildiz et al. (2016) propose a neural model which can pick out the best lemma and MSD analysis for each word in sentential context, but they rely on existing morphological analyzers to first produce the possible lemmata and MSDs.

Another vein of lemmatization is the **lemmatization of non-standard languages**. Kestemont et al. (2016) handles lemmatization of Middle Dutch with neural models. Specifically, they use a character-level CNN to learn token representations, and a neural encoder to extract character- and word-level features from a fixed-length token window of 2 words to the left and 2 words to the right, which are then used to predict the target lemma from a closed set of true lemmata. Manjavacas et al. (2019) take a **joint** learning approach to improve the lemmatization of non-standard historical languages, where the sentence encoder is trained jointly for lemmatization and **language modeling**. Specifically, they take lemmatization as a string transduction task with an encoder-decoder architecture and use a hierarchical sentence encoder similar to Kondratyuk et al. (2018) to enrich the encoder-decoder architecture with sentence context information. Their system does not require POS and MSD tagging. With the joint lemmatization and language modeling sentence encoder, their system can make use of both character-level and word-level features of each input token, a more general approach to use a predefined fixed-length window of neighbouring tokens, like in Kestemont et al. (2016) and Bergmanis and Goldwater (2018). They compared their system with a simple seq2seq model without sentence-level information, a model without joint language modeling loss and two state-of-the-art non-neural models using edit-tree induction, Morfette (Chrupala et al., 2008) and Lemming (Müller et al., 2015). Their system achieved state of the art performance on historical language lemmatization and comparable performance on standard languages. They found that the language modeling loss is helpful for capturing morphological information, and edit-tree-based approaches can be very effective for highly synthetic languages while the encoder-decoder architecture are more effective for languages with higher ambiguity and token-lemma ratio. The eight corpora of historical language they use are medieval and early modern languages, including Middle Dutch, Middle Low German, Medieval French, Historical Slovene and Medieval Latin, and they published the datasets.

5.3 Neural models and morphological tagging

Morphological tagging is usually conducted on words in sentences, making this a sequence labeling problem, where for each word in the sentence the tagger is expected to predict the corresponding MSD tag. This corresponds to the one-to-one learning scenario. The use of information from the context is critical for good tagging performance.

The typical neural model for this task has three stages: (1) word representation part, (2) context encoding part, (3) tagging part. Labeau et al. (2015) use a convolutional layer to derive word representations from a sequence of character embeddings, and compare the effect of using such character-based embeddings with using only conventional fixed-vocabulary word embeddings. They experimented with the feedforward architecture and bidirectional LSTM architecture for the context encoding part. For the tagging part, they experimented with two approaches: a simple softmax prediction, i.e. to apply a softmax function to the output at each time step from the context encoding to predict the corresponding tag for that time step; and a structured prediction approach, where they use the Viterbi algorithm (Viterbi, 1967) to infer the most likely MSD tag one feature after another. They evaluated their model with the German corpus TIGER Treebank (Brants et al., 2002). The best result was achieved by combining the character-based character embedding derived with a CNN layer and the conventional word embedding, using bidirectional LSTM to encode the context and applying the Viterbi algorithm to predict the tag sequence in a structured way. Their model outperformed the previous state-of-the-art model (Mueller et al., 2013) on the same dataset, which is a high-order CRF model with intensive feature engineering. Heigold et al. (2016) and Heigold et al. (2017) focus on evaluating different ways to get word representations on different languages, including feedforward, CNN, CNNHighway, LSTM and bidirectional LSTM. Their models use bidirectional LSTM to encode the context and a simple softmax layer applied to each time step to predict the tags. They achieved even better result. Heigold et al. (2016) found that when carefully tuned, the difference between the embedding models is minor, and Heigold et al. (2017) found that the LSTM-based approach is slightly better and more consistent than CNN-based approaches.

5.3.1 Cross-lingual transfer

Cotterell and Heigold (2017) apply the model in Heigold et al. (2017) for **cross-lingual transfer** learning. They train an RNN tagger for a low-resource language jointly with a tagger for a related high-resource language and the models for low- and high-resource languages share character-level features. They experimented with three cross-lingual transfer learning scheme: language-universal softmax, language-specific softmax, and joint morphological tagging and language identification. In the language-universal softmax scheme, they add a symbol as language identifier before and after the word and use the new form as input to the bidirectional LSTM embedding layer. In the language-specific softmax scheme, the model is trained on the combined data of all the languages, but has an independent output layer for every language. For the joint learning setting, the model predicts the language and MSD tag together. They found that the shared character representations among related languages successfully enable knowledge transfer between these languages. Their model, especially with the joint morphological tagging and language identification scheme, outperforms the MARMOT tagger (Mueller et al., 2013), and the tagger by Buys and Botha (2016). Buys and Botha (2016) also investigate morphological tagging for low-resource languages by cross-lingual transfer. The model they propose is a neural discriminative model based on Wsabie (Weston et al., 2011). Their approach is projection-based and the neural model learns to rank the set of tags allowed by the projection. The projection is conducted as a separate stage before the neural network modeling, and a separate morphological tagger is needed for the source-language text. In contrast, the approach of Cotterell and Heigold (2017) for cross-lingual transfer is end-to-end. Though compared to Cotterell and Heigold (2017) which still requires a small amount of data for the low-resource language with direct MSD annotation, Buys and Botha (2016) don't have this requirement, the parallel text between high-resource languages and low-resource languages can still be costly to get and the performance of the supervised morphological tagger for the source language is critical for obtaining high-quality constraints for the target language tagging.

Malaviya et al. (2018) point out that the model of Cotterell and Heigold (2017) has the shortcoming of only being able to output tags that have appeared in the training data and that the tagsets should be shared between the languages for the knowledge transfer. They propose a factorial conditional random field (Sutton et al., 2007) combined with bidirectional LSTM method which improves in these two aspects. Their approach achieved better performance than Cotterell and Heigold (2017). In addition, the graphical approach can model dependencies between tags for the whole unfactored MSDs and dependencies between unfactored MSDs in the sentence with better interpretability.

Kirov et al. (2017) is also about **cross-lingual transfer** effect, and it is also projection-based and requires parallel data as Buys and Botha (2016), but the research question is different: Kirov et al. (2017) aim at verifying the linguistic claim about equal expressiveness of languages, i.e. whether it's true that what is conveyed by syntactic or contextual cues in languages with comparatively limited morphology is expressed by overt inflection or derivation in languages with rich morphology. They first project the complex morphological tags of Czech words directly onto the English words they align to in the Prague Czech-English Dependency Treebank (PCEDT) (Hajič et al., 2012), and then develop a neural network tagging model with the feedforward architecture that takes input English features, including basic features derived directly from text, features derived from dependency trees and features from CFG parsing. The neural model attempts to classify the English word into the projected Czech tags. The high classification accuracy produced with the English features, gives support to the linguistic hypothesis of equal expressivity.

Conforti et al. (2018) is different from the tagging papers discussed above because they tag sequence of lemmata with part-of-speech and morphological features, rather than tag surface word forms. They presuppose the data has been lemmatized and use Lemming (Müller et al., 2015) to preprocess the data. Their tagging model is three layers of bidirectional GRU which takes the sequence of lemmata as input and predict the tags for each lemma with a softmax layer. They point out that their work can be helpful for machine translation.

5.3.2 Joint learning

In addition to **joint** training of morphological tagging and language identification (Cotterell and Heigold, 2017), it has been common to jointly model morphological tagging and lemmatization (specifically, lemmatization in context), which has been discussed in the previous subsection on context-sensitive lemmatization.

6 Discussion

6.1 Advantages of neural models

Neural network models have been very successful in processing morphology, and have improved the performance of almost all tasks in computational morphology by a large margin over previous state-of-the-art non-neural models. This success was especially impressive in SIGMORPHON 2016 shared task (Cotterell et al., 2016a) of morphological (re)inflection which received both neural and non-neural system submissions with all best-performed models being neural and the best neural model outperforming the best non-neural model by over 10% in average accuracy. Neural models have become dominant in all computational morphology tasks. Though neural models are data-hungry and require a large amount of labeled data to achieve good performance, later research on improving neural network models in low-resource scenarios also turn out to be useful and neural models with augmentations can still produce results better than or comparable to non-neural models when the training data is limited.

The success of deep learning methods is especially impressive with generation tasks in computational morphology. Before neural models became popular, work in computational morphology focused more on analysis rather than generation (Malouf, 2016; Malouf, 2017). Generation work before neural network includes Durrett and DeNero (2013).

Non-concatenative morphology like in Semitic languages has been a problem non-neural models can't handle well. Neural models have been very successful in this aspect (Wolf-Sonkin et al., 2018). In addition, neural models have been more flexible and creative, and thus can deal with unseen words or MSD tags as well as ambiguous words much more effectively (Bergmanis and Goldwater, 2018; Akyürek et al., 2019).

In contrast to the much better performance is the much less feature engineering. Finite state-machine based on rules written by human experts, which is very time-consuming and expensive (Beemer et al., 2020). Statistical machine learning approaches such as CRFs, SVMs usually require heavy feature engineering and heuristics inspired by language expert knowledge. Neural models usually don't need such feature engineering. In addition, they provide a way to replace traditional features in finite-state machine based approaches and statistical machine learning based approaches. For example, Rastogi et al. (2016) infuse features encoded with neural models into weighted FSTs for morphological inflection; graphical models like Cotterell and Heigold (2017), Wolf-Sonkin et al. (2018), and Vylomova et al. (2019) incorporate neural network approaches into statistical machine learning methods.

Another advantage of deep learning approaches is that they can incorporate contextual information and meaning (inferred from context, or using embedding information) more effectively. As shown in section 5, neural models for token-based tasks in computational morphology have achieved state-of-the-art performance, and tasks like morphological (re)inflection and lemmatization in context were rarely tackled before neural network models became popular for computational morphology.

Neural network models also make it possible to explore questions that are hard to explore before. For example, the cross-lingual transfer effect has been a phenomenon receiving wide research, thanks to neural network models. Can we use labeled data in high-resource languages to boost the model training for low-resource languages? How closely related should the high-resource language be in order to be helpful for the low-resource languages? These questions have been one of the focus of SIGMORPHON shared tasks (McCarthy et al., 2019) and received wide interest. Tagging tasks have seen improvements in models for low-resource languages when models are trained with related high-resource languages (e.g. Buys and Botha (2016), Cotterell and Heigold (2017), Akyürek et al. (2019), and Kondratyuk (2019)), while high-resource languages are found to contribute to the (re)inflection models for low-resource languages in some work (e.g. Kann et al. (2017b), Jin and Kann (2017), Anastasopoulos and Neubig (2019)), and don't in some others (e.g. Bergmanis et al. (2017), Rama and Çöltekin (2018), Çöltekin (2019), Hauer et al. (2019), Madsack and Weißgraeber (2019)). Though we still lack an effective evaluation method to determine whether the improvement in performance is actually transfer of linguistic knowledge between languages or just regularization or some other reasons, it's still exciting that neural models add this flexibility in using labeled data.

The end-to-end learning process of neural network models contributes to multi-task learning and joint learning in computational morphology. In addition, it provides an alternative way to model the acquisition of morphology, with the learning curve and result of neural approaches being similar to human language acquisition findings (Rumelhart and McClelland, 1986; Kirov and Cotterell, 2018), though the adequacy of neural networks as a good cognitive model is still questionable (Corkery et al., 2019).

6.2 Problems and future directions

It usually requires a good amount of examples in order to train a good neural network model. Liu and Hulden (2021) find that when measuring the amount of the training examples for morphological tasks, we should count the lemmata rather than inflected forms: a good representation of different lemmata is critical for the model to learn generalization over morphology.

In addition to being data-hungry, there are other problems with neural network approaches. One notorious problem that people have been trying to overcome, is the lack of interpretability. Current efforts to improve the interpretability in the learning of morphology is mainly to speculate what neural models have learned by error analysis of the output, ablation analysis of the model to see how much each part of the model contributes to the performance, plotting attention weights to see how much different parts of the input contribute at each step for generating the final prediction with the hope to find the correlation between the output and the parts attended to or plotting embedding representations with dimension reduction. Such interpretability alone can't contribute to better understanding of human language. More work needs to be done to interpret neural models as well as the understanding of human languages from the linguistics and cognitive science perspectives.

Related to model interpretability is the contribution of linguistic knowledge to computational processing of morphology. Though neural network approaches do not require high-level feature engineering, incorporating linguistically motivated knowledge has been found helpful for computational morphology tasks, e.g. Silfverberg et al. (2018b; Vylomova et al. (2019; Malaviya et al. (2019; Liu and Hulden (2020b)). It would be beneficial to develop tools facilitating linguistic studies as well if neural models can leverage and improve upon linguistic knowledge. This is a direction where more future work is necessary.

Morphology has an interface with syntax, which is reflected mainly in token-based computational morphology tasks, i.e. morphological processing of words in sentential context. This aspect has seen great progress with neural network approaches. Another interface of morphology with the language system is the interface between morphology and sounds, i.e. phonetics and phonology. Little work has been done with neural models as to allophonic and allomorphic analysis. This may be due to the lack of annotated data in this aspect, and labeling such data requires expert linguistic knowledge, especially on phonetics and phonology, making the labeling really expensive. With the end-to-end training fashion of neural models, perhaps some attempts can be conducted to use audio to assist the computational processing of this interface. In addition, though neural network models have been dealing with different writing systems well, even for learning cross-lingual information, in order to explore more about the interface between morphology and phonology, a phonemic representation of words rather than an orthographic one would be very helpful. Grapheme-to-phoneme (G2P) conversion systems (e.g. Ott et al. (2019)) can be used for this purpose.

Unsupervised learning was a large part of computational morphology before neural models (Goldsmith et al., 2017; Hammarström and Borin, 2011; Ruokolainen et al., 2016), which was primarily on segmentation. With neural network model, the work closely related to unsupervised word segmentation is on subword units. Bojanowski et al. (2017) use character n-grams to enrich word vectors, which turns out to be very effective, especially to deal with rare words. Sennrich et al. (2016) apply the byte pair encoding (BPE) algorithm (Gage, 1994) to get subword units, and show that segmentation with BPE improves over character n-gram models. BPE has been the commonly used segmentation method to get subword units for downstream processing with neural network models, though Liu et al. (2021) found that for morphologically complex language, segmentation by syllables has some advantage in machine translation. However, such unsupervised segmentation are not intended to learn segment boundaries that matches morpheme boundaries. The effort and use of neural models to learn actual morpheme boundaries in an unsupervised way is quite limited. More work can be done as to unsupervised learning of morphology with neural models in the future, especially considering the current success of deep learning approaches with token-based morphology tasks, which indicates that neural models can infer morphosyntactic information well from context.

There are other linguistic questions which can be explored more with neural models. One of such questions is about the equal expressiveness of languages, i.e. whether what is expressed in one language with morphology can be equally expressed in another language with syntactic or discourse techniques. Kirov et al. (2017) is a paper on this where they first project Czech MSD tags to aligned parallel English words in the PCEDT corpus, and develop a neural model predicts the complex MSD tags mapped from Czech to the English words. Their result support the linguistic hypothesis. Belinkov et al. (2017) train machine translation models between morphologicaly rich and morphologicaly poor languages, and test the model on morphological tasks. This is also a related effort. In addition, this question is closely related to the multilingual parsing task (Zeman et al., 2018). More work can be conducted in this direction, and multilingual parsing tasks can be a good reference.

7 Conclusion

Neural network approaches have been applied to computational morphology, including both generation and analysis tasks, with great success for concatenative as well as non-concatenative morphology systems. The current state-of-the-art architecture is the Transformer. Before the Transformer, the most commonly used neural network architecture is (bidirectional) RNN with attention. The two gated RNN architectures – LSTM and GRU – have comparable performance though LSTM have been adopted in relatively more work. The feedforward neural network is also commonly used in computational morphology. The use of CNN is quite limited, used mainly to get better word embedding representations from characters. For morphological generation tasks, the encoder-decoder architecture has been dominant. Before the Transformer was successfully applied, the RNN-based encoder-decoder model with soft attention mechanism produced state-of-the-art performance when the training data size is large. Hard attention mechanism leverages the fact that the transduction in morphological generation has more copying than in machine translation. The seq2seq model with hard attention can tackle the generation tasks better when the training data is limited and achieve performance as good as or even better than soft attention-enhanced seq2seq architecture in high-resource situations. The Transformer model has recently been applied to the morphological (re)inflection task, and produced the current state-of-the-art performance, even when the training data is limited (Wu et al., 2020; Vylomova et al., 2020; Liu and Hulden, 2020a).

The advantage of neural network approaches is not only much better performance with little high-level feature engineering. With these approaches, end-to-end learning becomes convenient, eliminating the error transmission between steps in the pipeline setting commonly used before deep learning methods. What's more, neural network approaches make multi-task and joint learning more doable. In addition, it becomes possible to model and explore research questions which are hard to study before, such as cross-lingual transfer and modeling of the non-linearity in human language acquisition, etc. Morphology in sentential context is also tackled more with better performance, thanks to neural network models which can represent context and meaning more effectively with dense vector representations.

Though data-hungriness has been a problem with neural network models, with recent efforts to improve neural models in low-resource settings and data augmentation techniques, neural models have achieved as good or even better performance when the training data is limited, compared to more traditional machine learning approaches.

However, the neural network approach is not without problems. The lack of interpretability has made it hard to contribute to the understanding of human language, which is one of the focus of computational morphology. Current effort to interpret neural models has been trying to explain what these models have learned, but how humans understand and process language may be different from these models. It is questionable whether being able to speculate what neural models learn can shed light on human language understanding and generation.

Another way neural models can contribute to the study of language, or morphology in the context of this paper, is to generate high-quality predictions even when the annotated data is limited, since the data in linguistic studies is usually limited compared to the large amount of data NLP tasks require. However, the performance of neural models in low-resource situations is not as good. In addition, the extreme situation of being low-resource is where there is no labeled data at all, a condition where we need to conduct unsupervised learning or reinforcement learning. There have been some unsupervised learning work in computational morphology before neural models, but little work has been done for unsupervised learning or reinforcement learning of morphology with neural models. Future work is needed to solve these problems.

References

Farrell Ackerman, James P. Blevins, and Robert Malouf. 2009. Parts and wholes: Implicative patterns in inflectional paradigms. In James P. Blevins and Juliette Blevins, editors, *Analogy in Grammar*, pages 54–82. Oxford University Press.

Judit Ács. 2018. BME-HAS system for CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 121–126, Brussels, October. Association for Computational Linguistics.

Željko Agić and Natalie Schluter. 2017. How (not) to train a dependency parser: The curious case of jackknifing part-of-speech taggers. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 679–684, Vancouver, Canada, July. Association for Computational Linguistics.

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada, July. Association for Computational Linguistics.

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 *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 41–48. Association for Computational Linguistics.

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 569–578, Gothenburg, Sweden, April. Association for Computational Linguistics.

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

Ekin Akyürek, Erenay Dayanık, and Deniz Yuret. 2019. Morphological analysis using a sequence decoder. *Transactions of the Association for Computational Linguistics*, 7:567–579.

Iñaki Alegria and Izaskun Etxeberria. 2016. EHU at the SIGMORPHON 2016 shared task. a simple proposal: Grapheme-to-phoneme for inflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 27–30, Berlin, Germany, August. Association for Computational Linguistics.

Antonios Anastasopoulos and Graham Neubig. 2019. Pushing the limits of low-resource morphological inflection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 984–996, Hong Kong, China, November. Association for Computational Linguistics.

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

Sarah Beemer, Zak Boston, April Bukoski, Daniel Chen, Princess Dickens, Andrew Gerlach, Torin Hopkins, Parth Anand Jawale, Chris Koski, Akanksha Malhotra, Piyush Mishra, Saliha Muradoglu, Lan Sang, Tyler Short, Sagarika Shreevastava, Elizabeth Spaulding, Testumichi Umada, Beilei Xiang, Changbing Yang, and Mans Hulden. 2020. Linguist vs. machine: Rapid development of finite-state morphological grammars. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 162–170, Online, July. Association for Computational Linguistics.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada, July. Association for Computational Linguistics.

Toms Bergmanis and Sharon Goldwater. 2018. Context sensitive neural lemmatization with lematus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1391–1400. Association for Computational Linguistics.

Toms Bergmanis and Sharon Goldwater. 2019. Training Data Augmentation for Context-Sensitive Neural Lemmatizer Using Inflection Tables and Raw Text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4119–4128, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 31–39, Vancouver, August. Association for Computational Linguistics.

Johannes Bjerva. 2017. One model to rule them all: multitask and multilingual modelling for lexical analysis. *arXiv preprint arXiv:1711.01100.*

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Marcel Bollmann and Anders Søgaard. 2016. Improving historical spelling normalization with bi-directional lstms and multi-task learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 131–139. The COLING 2016 Organizing Committee.

Marcel Bollmann, Joachim Bingel, and Anders Søgaard. 2017. Learning attention for historical text normalization by learning to pronounce. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 332–344, Vancouver, Canada, July. Association for Computational Linguistics.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168.

Jan Buys and Jan A. Botha. 2016. Cross-lingual morphological tagging for low-resource languages. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1954–1964, Berlin, Germany, August. Association for Computational Linguistics.

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

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume, and John Langford. 2015. Learning to search better than your teacher. In *International Conference on Machine Learning*, pages 2058–2066. PMLR.

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

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Grzegorz Chrupala, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with morfette. In *LREC 2008.*

Çağrı Çöltekin. 2019. Cross-lingual morphological inflection with explicit alignment. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 71–79, Florence, Italy, August. Association for Computational Linguistics.

Costanza Conforti, Matthias Huck, and Alexander Fraser. 2018. Neural morphological tagging of lemma sequences for machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 39–53, Boston, MA, March. Association for Machine Translation in the Americas.

Maria Corkery, Yevgen Matusevych, and Sharon Goldwater. 2019. Are we there yet? encoder-decoder neural networks as cognitive models of English past tense inflection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3868–3877, Florence, Italy, July. Association for Computational Linguistics.

Ryan Cotterell and Georg Heigold. 2017. Cross-lingual character-level neural morphological tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 748–759, Copenhagen, Denmark, September. Association for Computational Linguistics.

Ryan Cotterell and Hinrich Schütze. 2018. Joint semantic synthesis and morphological analysis of the derived word. *Transactions of the Association for Computational Linguistics*, 6:33–48.

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

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

Ryan Cotterell, Arun Kumar, and Hinrich Schütze. 2016b. Morphological segmentation inside-out. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2325–2330, Austin, Texas, November. Association for Computational Linguistics.

Ryan Cotterell, Tim Vieira, and Hinrich Schütze. 2016c. A joint model of orthography and morphological segmentation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 664–669. Association for Computational Linguistics.

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

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

Ryan Cotterell, John Sylak-Glassman, and Christo Kirov. 2017c. Neural graphical models over strings for principal parts morphological paradigm completion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 759–765. Association for Computational Linguistics.

Ryan Cotterell, Ekaterina Vylomova, Huda Khayrallah, Christo Kirov, and David Yarowsky. 2017d. Paradigm completion for derivational morphology. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 714–720, Copenhagen, Denmark, September. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, Mans Hulden, and Jason Eisner. 2018a. On the diachronic stability of irregularity in inflectional morphology. *arXiv preprint arXiv:1804.08262*.

Ryan Cotterell, Christo Kirov, Sebastian J. Mielke, and Jason Eisner. 2018b. Unsupervised disambiguation of syncretism in inflected lexicons. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 548–553, New Orleans, Louisiana, June. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Gėraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sebastian Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018c. The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 1–27. Association for Computational Linguistics.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Mathieu Dehouck and Pascal Denis. 2018. A framework for understanding the role of morphology in universal dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2864–2870, Brussels, Belgium, October-November. Association for Computational Linguistics.

Daniel Deutsch, John Hewitt, and Dan Roth. 2018. A distributional and orthographic aggregation model for English derivational morphology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1938–1947, Melbourne, Australia, July. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

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

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 634–643, San Diego, California, June. Association for Computational Linguistics.

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

Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420.

John A Goldsmith, Jackson L Lee, and Aris Xanthos. 2017. Computational learning of morphology. *Annual Review of Linguistics*, 3:85–106.

Alex Graves. 2008. Supervised sequence labelling with recurrent neural networks. *Ph. D. thesis*.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English dependency treebank 2.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3153–3160, Istanbul, Turkey, May. European Languages Resources Association (ELRA).

Harald Hammarström and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350.

Martin Haspelmath and Andrea Sims. 2013. *Understanding morphology*. Routledge.

Bradley Hauer, Amir Ahmad Habibi, Yixing Luan, Rashed Rubby Riyadh, and Grzegorz Kondrak. 2019. Cognate projection for low-resource inflection generation. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 6–11, Florence, Italy, August. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2016. Neural morphological tagging from characters for morphologically rich languages. *arXiv preprint arXiv:1606.06640*.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2017. An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 505–513, Valencia, Spain, April. Association for Computational Linguistics.

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. 1995. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161.

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

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2020a. DagoBERT: Generating derivational morphology with a pretrained language model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3848–3861, Online, November. Association for Computational Linguistics.

Valentin Hofmann, Hinrich Schütze, and Janet Pierrehumbert. 2020b. A graph auto-encoder model of derivational morphology. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1127–1138, Online, July. Association for Computational Linguistics.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Mans Hulden. 2014. Generalizing inflection tables into paradigms with finite state operations. In *Proceedings of the 2014 Joint Meeting of SIGMORPHON and SIGFSM*, pages 29–36. Association for Computational Linguistics.

Huiming Jin and Katharina Kann. 2017. Exploring cross-lingual transfer of morphological knowledge in sequence-to-sequence models. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 70–75.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2017. An exploration of neural sequence-to-sequence architectures for automatic post-editing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 120–129, Taipei, Taiwan, November. Asian Federation of Natural Language Processing.

Katharina Kann and Hinrich Schütze. 2016a. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 62–70. Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2016b. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 555–560, Berlin, Germany, August. Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2017a. The LMU system for the CoNLL-SIGMORPHON 2017 shared task on universal morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 40–48, Vancouver, August. Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2017b. Unlabeled data for morphological generation with character-based sequence-to-sequence models. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 76–81, Copenhagen, Denmark, September. Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2018. Neural transductive learning and beyond: Morphological generation in the minimal-resource setting. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3254–3264, Brussels, Belgium, October-November. Association for Computational Linguistics.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural morphological analysis: Encoding-decoding canonical segments. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 961–967. Association for Computational Linguistics.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017a. Neural multi-source morphological reinflection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 514–524, Valencia, Spain, April. Association for Computational Linguistics.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017b. One-shot neural cross-lingual transfer for paradigm completion. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1993–2003. Association for Computational Linguistics.

Katharina Kann, Jesus Manuel Mager Hois, Ivan Vladimir Meza Ruiz, and Hinrich Schütze. 2018. Fortification of neural morphological segmentation models for polysynthetic minimal-resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 47–57. Association for Computational Linguistics.

Yova Kementchedjhieva, Johannes Bjerva, and Isabelle Augenstein. 2018. Copenhagen at CoNLL–SIGMORPHON 2018: Multilingual inflection in context with explicit morphosyntactic decoding. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 93–98, Brussels, October. Association for Computational Linguistics.

Mike Kestemont, Guy De Pauw, Renske van Nie, and Walter Daelemans. 2016. Lemmatization for variation-rich languages using deep learning. *Digital Scholarship in the Humanities*, 32(4):797–815.

David King. 2016. Evaluating sequence alignment for learning inflectional morphology. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 49–53, Berlin, Germany, August. Association for Computational Linguistics.

Christo Kirov and Ryan Cotterell. 2018. Recurrent neural networks in linguistic theory: Revisiting pinker and prince (1988) and the past tense debate. *Transactions of the Association for Computational Linguistics*, 6:651–665.

Christo Kirov, John Sylak-Glassman, Rebecca Knowles, Ryan Cotterell, and Matt Post. 2017. A rich morphological tagger for English: Exploring the cross-linguistic tradeoff between morphology and syntax. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 112–117, Valencia, Spain, April. Association for Computational Linguistics.

Christo Kirov, Ryan Cotterell, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sebastian J. Mielke, Arya McCarthy, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. Unimorph 2.0: Universal morphology. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA).

Stav Klein and Reut Tsarfaty. 2020. Getting the ##life out of living: How adequate are word-pieces for modelling complex morphology? In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 204–209, Online, July. Association for Computational Linguistics.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing universal dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China, November. Association for Computational Linguistics.

Daniel Kondratyuk, Tomáš Gavenčiak, Milan Straka, and Jan Hajič. 2018. LemmaTag: Jointly tagging and lemmatizing for morphologically rich languages with BRNNs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4921–4928, Brussels, Belgium, October-November. Association for Computational Linguistics.

Dan Kondratyuk. 2019. Cross-lingual lemmatization and morphology tagging with two-stage multilingual BERT fine-tuning. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 12–18, Florence, Italy, August. Association for Computational Linguistics.

Natalia Korchagina. 2017. Normalizing medieval german texts: from rules to deep learning. In *Proceedings of the NoDaLiDa 2017 Workshop on Processing Historical Language*, pages 12–17. Linköping University Electronic Press.

Matthieu Labeau, Kevin Löser, and Alexandre Allauzen. 2015. Non-lexical neural architecture for fine-grained POS tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 232–237, Lisbon, Portugal, September. Association for Computational Linguistics.

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Ling Liu and Mans Hulden. 2017. Evaluation of finite state morphological analyzers based on paradigm extraction from Wiktionary. In *Proceedings of the 13th International Conference on Finite State Methods and Natural Language Processing (FSMNLP 2017)*, pages 69–74, Umeå, Sweden, September. Association for Computational Linguistics.

Ling Liu and Mans Hulden. 2020a. Analogy models for neural word inflection. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2861–2878, Barcelona, Spain (Online), December. International Committee on Computational Linguistics.

Ling Liu and Mans Hulden. 2020b. Leveraging principal parts for morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 153–161, Online, July. Association for Computational Linguistics.

Ling Liu and Mans Hulden. 2021. Can a transformer pass the wug test? tuning copying bias in neural morphological inflection models. *arXiv preprint arXiv:2104.06483*.

Ling Liu and Lingshuang Jack Mao. 2016. Morphological reinflection with conditional random fields and unsupervised features. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 36–40, Berlin, Germany, August. Association for Computational Linguistics.

Ling Liu, Ilamvazhuthy Subbiah, Adam Wiemerslage, Jonathan Lilley, and Sarah Moeller. 2018. Morphological reinflection in context: CU boulder's submission to CoNLL–SIGMORPHON 2018 shared task. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 86–92, Brussels, October. Association for Computational Linguistics.

Ling Liu, Zach Ryan, and Mans Hulden. 2021. The usefulness of bibles in low-resource machine translation. In *Proceedings of the 4th Workshop on the Use of Computational Methods in the Study of Endangered Languages (ComputEL-4)*. Association for Computational Linguistics.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Andreas Madsack and Robert Weißgraeber. 2019. AX semantics' submission to the SIGMORPHON 2019 shared task. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 1–5, Florence, Italy, August. Association for Computational Linguistics.

Peter Makarov and Simon Clematide. 2018a. Imitation learning for neural morphological string transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2877–2882. Association for Computational Linguistics.

Peter Makarov and Simon Clematide. 2018b. Neural transition-based string transduction for limited-resource setting in morphology. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 83–93. Association for Computational Linguistics.

Peter Makarov and Simon Clematide. 2018c. UZH at CoNLL-SIGMORPHON 2018 shared task on universal morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 69–75. Association for Computational Linguistics.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and copy: UZH at SIGMORPHON 2017 shared task for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 49–57, Vancouver, August. Association for Computational Linguistics.

Chaitanya Malaviya, Matthew R. Gormley, and Graham Neubig. 2018. Neural factor graph models for cross-lingual morphological tagging. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2653–2663, Melbourne, Australia, July. Association for Computational Linguistics.

Chaitanya Malaviya, Shijie Wu, and Ryan Cotterell. 2019. A simple joint model for improved contextual neural lemmatization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1517–1528, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Robert Malouf. 2016. Generating morphological paradigms with a recurrent neural network. *San Diego Linguistic Papers*, 6:122–129.

Robert Malouf. 2017. Abstractive morphological learning with a recurrent neural network. *Morphology*, 27:431–458.

Enrique Manjavacas, Ákos Kádár, and Mike Kestemont. 2019. Improving lemmatization of non-standard languages with joint learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1493–1503, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Jonathan May and Kevin Knight. 2006. A better n-best list: Practical determinization of weighted finite tree automata. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 351–358, New York City, USA, June. Association for Computational Linguistics.

Arya D. McCarthy, Ekaterina Vylomova, Shijie Wu, Chaitanya Malaviya, Lawrence Wolf-Sonkin, Garrett Nicolai, Christo Kirov, Miikka Silfverberg, Sebastian J. Mielke, Jeffrey Heinz, Ryan Cotterell, and Mans Hulden. 2019. The SIGMORPHON 2019 shared task: Morphological analysis in context and cross-lingual transfer for inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–244, Florence, Italy, August. Association for Computational Linguistics.

Arya D. McCarthy, Adina Williams, Shijia Liu, David Yarowsky, and Ryan Cotterell. 2020. Measuring the similarity of grammatical gender systems by comparing partitions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5664–5675, Online, November. Association for Computational Linguistics.

Sarah Moeller and Mans Hulden. 2018. Automatic glossing in a low-resource setting for language documentation. In *Proceedings of the Workshop on Computational Modeling of Polysynthetic Languages*, pages 84–93, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Sarah Moeller, Ling Liu, Changbing Yang, Katharina Kann, and Mans Hulden. 2020. IGT2P: From interlinear glossed texts to paradigms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5251–5262, Online, November. Association for Computational Linguistics.

Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA, October. Association for Computational Linguistics.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274. Association for Computational Linguistics.

Garrett Nicolai and Grzegorz Kondrak. 2017. Morphological analysis without expert annotation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 211–216, Valencia, Spain, April. Association for Computational Linguistics.

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

Garrett Nicolai, Bradley Hauer, Adam St Arnaud, and Grzegorz Kondrak. 2016. Morphological reinflection via discriminative string transduction. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 31–35, Berlin, Germany, August. Association for Computational Linguistics.

Garrett Nicolai, Bradley Hauer, Mohammad Motallebi, Saeed Najafi, and Grzegorz Kondrak. 2017. If you can't beat them, join them: the university of alberta system description. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 79–84, Vancouver, August. Association for Computational Linguistics.

Garrett Nicolai, Saeed Najafi, and Grzegorz Kondrak. 2018. String transduction with target language models and insertion handling. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 43–53, Brussels, Belgium, October. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

Robert Östling. 2016. Morphological reinflection with convolutional neural networks. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 23–26, Berlin, Germany, August. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Ben Peters and André F. T. Martins. 2019. IT–IST at the SIGMORPHON 2019 shared task: Sparse two-headed models for inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 50–56, Florence, Italy, August. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June. Association for Computational Linguistics.

Joël Plisson, Nada Lavrac, Dunja Mladenic, et al. 2004. A rule based approach to word lemmatization. In *Proceedings of IS*, volume 3, pages 83–86.

Taraka Rama and Çağrı Çöltekin. 2018. Tübingen-oslo system at SIGMORPHON shared task on morphological inflection. a multi-tasking multilingual sequence to sequence model. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 112–115, Brussels, October. Association for Computational Linguistics.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633. Association for Computational Linguistics.

Joana Ribeiro, Shashi Narayan, Shay B. Cohen, and Xavier Carreras. 2018. Local string transduction as sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1360–1371, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Alexander Robertson and Sharon Goldwater. 2018. Evaluating historical text normalization systems: How well do they generalize? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 720–725. Association for Computational Linguistics.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings.

David E Rumelhart and James L McClelland. 1986. On learning the past tenses of english verbs. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 2:216–271.

Teemu Ruokolainen, Oskar Kohonen, Kairit Sirts, Stig-Arne Grönroos, Mikko Kurimo, and Sami Virpioja. 2016. A comparative study of minimally supervised morphological segmentation. *Computational Linguistics*, 42(1):91–120, March.

Tatyana Ruzsics and Tanja Samardzic. 2017. Neural sequence-to-sequence learning of internal word structure. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 184–194. Association for Computational Linguistics.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

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

Uygun Shadikhodjaev and Jae Sung Lee. 2019. CBNU system for SIGMORPHON 2019 shared task 2: a pipeline model. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 19–24, Florence, Italy, August. Association for Computational Linguistics.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany, August. Association for Computational Linguistics.

Miikka Silfverberg and Mans Hulden. 2018. An encoder-decoder approach to the paradigm cell filling problem. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2883–2889, Brussels, Belgium, October-November. Association for Computational Linguistics.

Miikka Silfverberg, Adam Wiemerslage, Ling Liu, and Lingshuang Jack Mao. 2017. Data augmentation for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 90–99, Vancouver, August. Association for Computational Linguistics.

Miikka Silfverberg, Ling Liu, and Mans Hulden. 2018a. A computational model for the linguistic notion of morphological paradigm. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1615–1626, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Miikka P. Silfverberg, Lingshuang Mao, and Mans Hulden. 2018b. Sound analogies with phoneme embeddings. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2018*, pages 136–144.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 787–794, Sydney, Australia, July. Association for Computational Linguistics.

Benjamin Snyder and Regina Barzilay. 2008. Cross-lingual propagation for morphological analysis. In *AAAI*, pages 848–854.

Alexey Sorokin. 2016. Using longest common subsequence and character models to predict word forms. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 54–61, Berlin, Germany, August. Association for Computational Linguistics.

Alexey Sorokin. 2018. What can we gain from language models for morphological inflection? In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 99–104, Brussels, October. Association for Computational Linguistics.

Petra Steiner. 2019. Augmenting a German morphological database by data-intense methods. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 178–188, Florence, Italy, August. Association for Computational Linguistics.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada, August. Association for Computational Linguistics.

Milan Straka, Jana Straková, and Jan Hajic. 2019. UDPipe at SIGMORPHON 2019: Contextualized embeddings, regularization with morphological categories, corpora merging. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 95–103, Florence, Italy, August. Association for Computational Linguistics.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium, October. Association for Computational Linguistics.

Gregory T Stump. 2001. *Inflectional morphology: A theory of paradigm structure*, volume 93. Cambridge University Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8(3).

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 *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 71–75, Berlin, Germany, August. Association for Computational Linguistics.

Clara Vania, Andreas Grivas, and Adam Lopez. 2018. What do character-level models learn about morphology? the case of dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2573–2583, Brussels, Belgium, October-November. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

Ekaterina Vylomova, Ryan Cotterell, Timothy Baldwin, and Trevor Cohn. 2017. Context-aware prediction of derivational word-forms. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 118–124, Valencia, Spain, April. Association for Computational Linguistics.

Ekaterina Vylomova, Ryan Cotterell, Trevor Cohn, Timothy Baldwin, and Jason Eisner. 2019. Contextualization of morphological inflection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2018–2024, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Ekaterina Vylomova, Jennifer White, Elizabeth Salesky, Sabrina J. Mielke, Shijie Wu, Edoardo Maria Ponti, Rowan Hall Maudslay, Ran Zmigrod, Josef Valvoda, Svetlana Toldova, Francis Tyers, Elena Klyachko, Ilya Yegorov, Natalia Krizhanovsky, Paula Czarnowska, Irene Nikkarinen, Andrew Krizhanovsky, Tiago Pimentel, Lucas Torroba Hennigen, Christo Kirov, Garrett Nicolai, Adina Williams, Antonios Anastasopoulos, Hilaria Cruz, Eleanor

Chodroff, Ryan Cotterell, Miikka Silfverberg, and Mans Hulden. 2020. SIGMORPHON 2020 shared task 0: Typologically diverse morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 1–39, Online, July. Association for Computational Linguistics.

Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. 2016. Morphological segmentation with window LSTM neural networks. In *AAAI*, pages 2842–2848.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Adina Williams, Damian Blasi, Lawrence Wolf-Sonkin, Hanna Wallach, and Ryan Cotterell. 2019. Quantifying the semantic core of gender systems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5734–5739, Hong Kong, China, November. Association for Computational Linguistics.

Lawrence Wolf-Sonkin, Jason Naradowsky, Sebastian J. Mielke, and Ryan Cotterell. 2018. A structured variational autoencoder for contextual morphological inflection. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2631–2641. Association for Computational Linguistics.

Shijie Wu and Ryan Cotterell. 2019. Exact hard monotonic attention for character-level transduction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1530–1537, Florence, Italy, July. Association for Computational Linguistics.

Shijie Wu, Pamela Shapiro, and Ryan Cotterell. 2018. Hard non-monotonic attention for character-level transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4425–4438, Brussels, Belgium, October-November. Association for Computational Linguistics.

Shijie Wu, Ryan Cotterell, and Mans Hulden. 2020. Applying the transformer to character-level transduction. *arXiv preprint arXiv:2005.10213*.

Eray Yildiz and A. Cüneyd Tantuğ. 2019. Morpheus: A neural network for jointly learning contextual lemmatization and morphological tagging. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 25–34, Florence, Italy, August. Association for Computational Linguistics.

Eray Yildiz, Caglar Tirkaz, H Bahadır Sahin, Mustafa Tolga Eren, and Omer Ozan Sonmez. 2016. A morphology-aware network for morphological disambiguation. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1307–1316. Association for Computational Linguistics.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium, October. Association for Computational Linguistics.

Chunting Zhou and Graham Neubig. 2017a. Morphological inflection generation with multi-space variational encoder-decoders. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 58–65, Vancouver, August. Association for Computational Linguistics.

Chunting Zhou and Graham Neubig. 2017b. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 310–320. Association for Computational Linguistics.