Eberhard Karls Universität Tübingen
Seminar für Sprachwissenschaft

Neural Sequence to Sequence Lemmatization

Ohnomore!$_{seq2seq}$

Bachelorthesis

presented to the examination office
of the philosophical faculty

Author: Tobias Pütz

Supervisor: Dr. Daniël de Kok

Contents

Abstract

In this thesis, we explore lemmatization of Dutch and German using neural Sequence to Sequence (Seq2Seq) architectures. In particular, we compare attentional to non attentional architectures, investigate the combination of the decoder with a linear-chain Conditional Random Field and examine the influence of explicit morphological information fed to the decoder. We find that the non-morphological Seq2Seq models perform worse than existing systems that explicitly model morphological information. We show that supplying the Seq2Seq model with morphological information leads to competitive performance and provide insight in qualitative differences between a strong baseline and our proposed architecture.

1 Introduction

The process of mapping an inflected form to its dictionary entry, the lemma, is called lemmatization. In Natural Language Processing knowing the lemma of a form has an undeniable influence on downstream applications. Lemmas are common features for a wide range of tasks and have been shown to improve results in parsing (Dozat and Manning, 2018) and machine translation (Sennrich and Haddow, 2016). Apart from their application as input features to statistical classifiers, lemmas are a requirement to map any inflected form to a lexical resource. In Information Retrieval, for example, the goal is to present useful information given a search query. Suppose the query "car", in this case we do not only want results containing the exact word but we also want to find resources containing the word "cars". Lemmatization enables such a lookup as articles containing "cars" can be indexed with the lemma "car". If we are interested in distributional similarity a common approach is to algorithmically obtain a vector representing a certain word. A simple approach, which obtains discrete representations, is to explicitly count how often a certain word co-occurs with others, a technique called Pointwise Mutual Information (PMI). Word embeddings, in contrast to PMI, are continuous. They are obtained by maximizing the similarity between a vector representing the word and its context. Both are known to suffer from bad representations for rare words. This problem is especially relevant for morphologically rich languages where the occurences of rare words are divided between their possibly numerous inflections. Building the word representations based on lemmas leads to less sparse representations as the inflections of a word are seen as the same symbol, summing up their occurences.

Some existing lemmatization systems employ transformational and derivational rules in combination with lexica (Kruase and Willée, 1981; Schmid et al., 2004). These are known to handle out-of-vocabulary words poorly. Others enrich the input forms with linguistic information and then choose the correct transformation to the according lemma (Chrupała, 2006). Later a synergy between assigning this linguistic information and the lemma jointly was found (Chrupała et al., 2008; Müller et al., 2015). All these aforementioned approaches rely on extensive, sometimes language-specific, sets of handcrafted features.

More recently domain-agnostic neural Sequence to Sequence (Seq2Seq) architectures (Sutskever et al., 2014) have achieved impressive results in several sequence transduction tasks including machine translation (Sutskever et al., 2014; Cho et al., 2014a; Bahdanau et al., 2014), text summarization (See et al., 2017), constituency parsing (Vinyals et al., 2015) and, closely related to lemmatization, morphological reinflection (Cotterell et al., 2016, 2017). Schnober et al. (2016) and Bergmanis and Goldwater (2018) applied the Seq2Seq architecture to lemmatization. In the same vein, we also treat lemmatization as a string transduction task within the Seq2Seq framework. The task is then to generate the sequence of characters of the lemma $y_0, ..., y_n$ given the sequence of characters of the form $x_0, ..., x_n$ and possibly disambiguating information. More formally we aim to model $P(y_0, ..., y_n | x_0, ..., x_n; a)$ where a is a set of arbitrary features like the form's part-of-speech or morphological tags.

We evaluate several Seq2Seq architectures on both Dutch and German. The basis are those presented in Sutskever et al. (2014) and Luong et al. (2015). We propose varieties, namely combining the decoder with a Conditional Random Field (CRF), the usage of linear time attention Raffel et al. (2017) and providing the decoder with explicit morphological information. The main contributions of this thesis are:

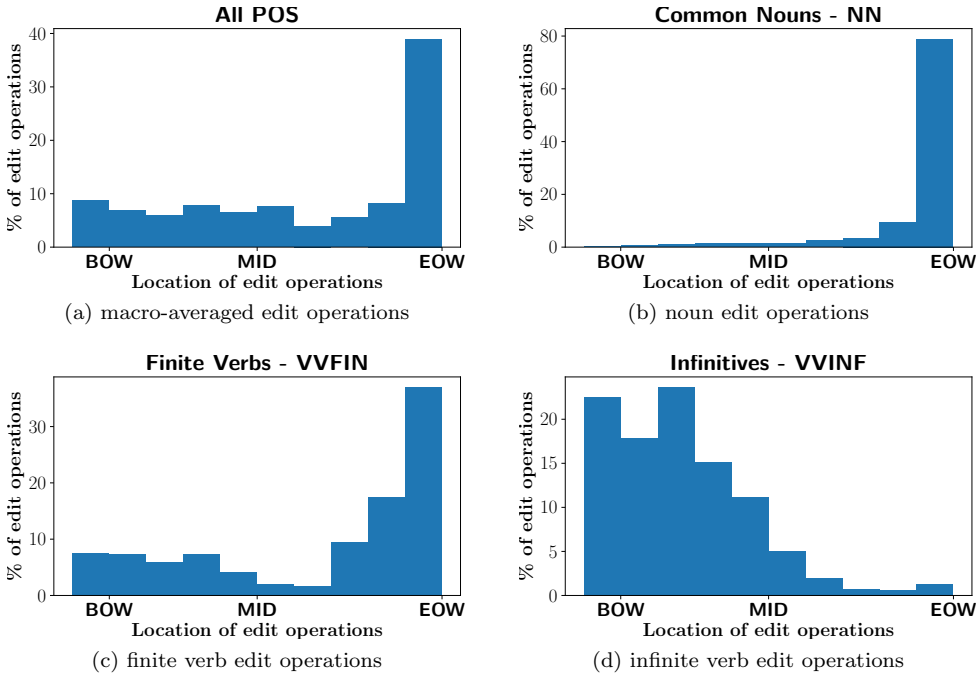
(i) a thorough evaluation of several Seq2Seq architectures on lemmatizing German,

(ii) the exploration of employing a Conditional-Random-Field within the Seq2Seq framework

(iii) the investigation of the effect of providing explicit morphological information to a Seq2Seq model.

The rest of this thesis will be structured as follows, first, we will discuss several challenges faced when lemmatizing German, then we will give an overview of existing approaches to lemmatization and relevant learning methods. After going through the related work we will introduce the several Seq2Seq variations we propose. Next we will discuss the evaluation methods and introduce the datasets. Following this we will show the results of the evaluation, first quantitatively, then we will provide likely causes of differences in performance between the several architectures. Finally we will close with a discussion of the work and an outlook to future work.

2 Morphological Challenges



(a) macro-averaged edit operations

(b) noun edit operations

(c) finite verb edit operations

(d) infinite verb edit operations

Figure 1: Locations of edit operations to transform forms to their lemma in TüBa-D/Z.

In this section we will discuss morphological challenges in German lemmatization. As can be seen in Figure 1, German morphology is mostly limited to the suffix. Most verbs also follow this pattern, however, certain forms introduce prefixes like in "ge-schlossen" (closed) which is the past participle of "schließen" (to close). Forms that follow paradigms like "schließen", where the inflection results in a change within the word root, are considered to be irregular. Irregular inflections are not limited to verbs but also occur, less frequently, in certain adjectives like "gut" (good) that have the comparative "besser" (better) and the superlative "am besten" (best).

Another important group of verbs, separable verbs, like "abschließen", have a prefix, which can appear separately, as demonstrated in (1).

(1) Als er erfuhr, dass er die Tür **abschließen** soll, **schloss** er sie
When he heard, that he the door lock should, locked he it
ab.
(prefix-ab).

Separable verbs have special infinitives and past participles. They are some of the rare accounts of concatenative morphology in German, where the prepositional prefix is followed by the infix "-zu-", the German infinitive marker, or "-ge-", a participle marker. (2) and (3) illustrate this.

(2) Als er erfur, dass er die Tür **abzuschließen** hatte, **schloss** er sie **ab**.
When he heard, that he the door lock has-to, locked he it (prefix-ab).

(3) Er hat die Tür **abgeschlossen**.
He did the door lock.

Since the same verbal root can occur with multiple prefixes ("auf-schließen", "ab-schließen"), some treebanks, like TüBa-D/Z (Telljohann et al., 2004), treat the root as the lemma, disregarding the prefix. The difficulty with prefixed verbs is to determine which of the possibly multiple prefixes is separable. Some prefixes can always be separated, some never, but for some there exist verb forms where both is possible. In (4) and (5) it becomes even harder as "geleitet" is an ambiguous homograph. In (4) it is the past participle of the verb "leiten", therefore "ge-" is a separable prefix, in (5), however, it is simple present third person of the verb "geleiten".

(4) Sie hat das Unternehmen **geleitet**
She did the company lead.

(5) Sie **geleitet** ihn in Sicherheit.
She lead him to safety.

In German, infinitives are usually the lemma of a verb, due to this property we can use them to quantify separable verbs in Figure 1-d, as the only infinitives that are not a lemma are those with a separable prefix.

As with most natural languages, German morphology also exhibits ambiguities in the form of syncretisms. Several syncretisms can be found with animate noun phrases. Some German nouns with the masculine singular ending "-er" and the feminine "-in", like "Schauspieler" (actor) and "Schauspielerin" (actress) have no marked nominative plural for the masculine form. Others, like "Vorsitzenden" in (6), do not mark the gender in the nominative plural.

(6) Die Vorsitzenden trafen sich zum Krisengespräch.
The chair-(wo)men met (refl) for the crisis meeting.

Müller et al. (2015) argue that it is important to know the lemma of these forms in order to assign a gender. We argue that in some cases the singular form can only be recognized if, possibly extra-sentential, discourse information is available. In other cases, e.g. if a plural word describes a mixed gendered group of people, the word cannot be reduced to a singular form since no uni-gender form exists. Hence we should consider the plural form as the lemma.

As Kruase and Willée (1981) point out, it is not only necessary to disambiguate between morphemes, but also to decide which part of the suffix is inflectional. An example is "Rasen" (lawn) and "Steinen" (stones). The former "-en" ending is not inflectional, while the latter marks dative plural. In German, the "-en" suffix is responsible for quite some ambiguity. Besides being part of the spelling or a morpheme, it is sometimes only partly inflectional. In "Lampen" (lamps), for example, "-e-" is part of the spelling while "-n" is a plural marker. One class of German verb nominalization adds to this kind of ambiguity. The words of this class only differ from the infinitive they are derived from by having a capitalized first letter and an accompanying determiner. All of them have no plural and follow the neuter declension class. In many cases these infinitives end in "-en".

(7) Bei diesen Träumen, sollte er das
with these(dat.pl) dreams(dat.pl), should he the(acc.sg)
Träumen lieber bleiben lassen.
dreaming(acc.sg) rather be let.

In (7), "Träumen" has two readings: (i) dative plural of "Traum" (dream) and (ii) accusative singular of the nominalized verb "träumen" (to dream). Since German grammar enforces gender, case and number congruence, we can use the morphological information of the neighbouring determiners "diesen" and "das" to determine whether "Träumen" is a nominalized verb or a declensed noun.

In the previous example we argued that we can leverage the morphological information coded into the two determiners to disambiguate between the two readings of "Träumen". If we take a closer look at these determiners, we will notice that they are also highly ambiguous. The demonstrative pronoun "diesen", for instance, can have four different morphological analyses: dative plural for all three grammatical genders: feminine, masculine and neuter and also accusative singular masculine. For each of the genders a different lemma exists. As the preposition "bei" sanctions a dative, we can infer the case in (7). The gender can be resolved as we know that nominalized verbs of this class have no plural, hence the number congruence requirement informs us that reading it as a nominalization is ungrammatical.

Together with prepositions, conjunctions and the like, determiners and demonstrative pronouns belong to the group of closed class parts-of-speech. Even though some of them are inflected and, as shown before, highly ambiguous, most of them have no inflected forms.

3 Related Work

In the following section we will describe work in the fields of lemmatization, stemming and morphological reinflection as well as general learning methods relevant to lemmatization and this work.

3.1 Rule Based Lemmatization

Kruase and Willée (1981) is an early account of rule-based German lemmatization. Their approach, much shaped by the technology of the time, aims to be efficient. It relies on multiple fallbacks. First, a full-form dictionary is queried. If no entry was found, they determine an analysis routine based on the suffix of a word. The according routine then performs several edit operations coupled with dictionary lookups, if no matches are returned, possible affixes are eliminated and the result is returned as the lemma. They also employ a set of 12 context-sensitive rules to disambiguate the part-of-speech.

3.1.1 Stemming

Stemming used to be a popular method in Information Retrieval. The goal here, much in the same vein as lemmatization, is to reduce possibly inflected forms through edit operations to a common form. The most prominent stemming tool is the Snowball Stemmer (Porter, 2001), which formalizes an algorithm that produces the stem of a word by applying iterative transformational rules. Implementations are available for many languages, including English (Porter, 1980).

The most important distinction between lemmatization and stemming is that in stemming the reduction to a common form is not guaranteed to be a valid word, the only goal is to reduce inflected forms to a common representation. Table 1 illustrates the differences between stemming and lemmatization.

Form	Stem	Lemma
schlugen	schl	schlagen
beaten	beat	beat

Table 1: Differences between stemming and lemmatization

3.1.2 Finite State Transducers

Finite State Transducers (FST) are a popular method for morphological analysis (Minnen et al., 2001; Schmid et al., 2004; Sennrich and Kunz, 2014). An FST morphological analyzer commonly consists of a lexicon containing words and morphemes and a set of rules describing how words and morphemes can be combined. Lemmatization is often only one of the tasks they solve. Given enough expert effort, these tools achieve very good coverage. However, as most

non-statistical tools, they are often not able to disambiguate and can only output all possible analyses of a given input.

3.2 Linear Classifiers Choosing Edit-Scripts

Chrupała (2006) were the first to cast lemmatization as a classification task. During training, they derived the shortest edit script (SES) to transform a form into its lemma and then learned to predict the correct edit script using a Support Vector Machine (SVM). Chrupała et al. (2008) extended this with their Morfette system and started to jointly assign morphological tags and edit scripts using Maximum Entropy Classification.

3.3 Conditional Random Fields

Conditional Random Fields (CRF) (Lafferty et al., 2001) have been proven to perform exceptionally well on spatial modelling tasks. They combine the strength of a discrete local classifier with a graphical model in a way that allows for arbitrary features while taking transition probabilities between states into account.

In Natural Language Processing, CRFs found application in named entity recognition (NER), POS-tagging, morphological tagging and speech recognition (Sutton and McCallum, 2010; Müller et al., 2013).

In NER Lample et al. (2016) combined CRFs with another powerful sequence learner, namely Recurrent Neural Networks, which will be discussed in the following section.

Despite their effectiveness, CRFs, especially of higher order, had not been applied to tasks with a big label vocabulary, mostly due to their runtime, which is in $O(N^n T)$ where N is the number of output states, n the order of the CRF and T the sequence length (Müller, 2015). Müller et al. (2013) addressed this problem with their tagger MarMoT by introducing a pruning strategy which incrementally builds up higher order CRFs from lower order ones and limits the search space by dropping low probability sequences.

In lemmatization CRFs became relevant as Müller et al. (2015), further optimizing the approach of Chrupała et al. (2008), achieved state-of-the-art performance in lemmatizing several languages. They used their pruned CRF-tagger MarMoT to assign POS- and morphological tags to the input of the Lemming system. Lemming also casts lemmatization as a classification task, where the goal is to choose an edit tree which transforms a form to its lemma.

Lemming and MarMoT can be trained separately or jointly. While Müller et al. (2015) report improved performance by training jointly, it also forced them to evaluate on tokens instead of types which we expect to bias their evaluation towards frequent tokens, that we also suspect to appear both in training and validation set.

3.4 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a special case of Artificial Neural Networks (ANN). In contrast to simple ANNs that process each input independently, RNNs are able to process sequences of discrete symbols in a way that takes previously processed symbols into account. Due to their stateful properties, they are an adequate tool to deal with non-local dependencies as often encountered in natural language.

Analogous to multi-layered ANNs, where the output of a layer is a function of all previous layers, RNNs can be described as follows:

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

where h_t denotes the state of the RNN at time t and f some compositional nonlinear function combining x_t, the input at t, and the previous state h_{t-1}. The common training procedure of such networks is a technique called backpropagation through time (BPTT) (Rumelhart et al., 1985; Werbos, 1990). BPTT exploits that any RNN with a finite input sequence can be reformulated as a deep feed-forward network where the weights are fixed for all layers.

$$h_t = f(f(f(f(h_{init}, x_0), x_{...}), x_{t-1}), x_t) \tag{2}$$

Equation 2 rewrites Equation 1 as such a sequence of nested functions. BPTT uses this formulation to calculate updates of the network's weights. The update for a certain parameter is then obtained by applying the chain rule to calculate the partial derivative of some loss function with respect to that parameter.

Early RNNs like Simple RNNs (SRN) (Elman, 1990) suffered from unreliable and unstable learning caused by the chained multiplications of the gradients. Multiplying many gradients smaller than 1 together effectively causes the gradient to become 0, the vanishing gradient problem. The opposite case, the exploding gradient problem, describes the scenario when multiple gradients are bigger than 1 (Bengio et al., 1994; Pascanu et al., 2012). Numerous attempts to deal with these problems have been made both by changing the learning method and system architectures until Hochreiter and Schmidhuber (1997) came up with the Long Short-Term Memory architecture (LSTM), effectively solving the vanishing gradient problem by introducing several gating mechanisms that control the gradient flow. Pascanu et al. (2012) argue that LSTM does not solve the exploding gradient and only circumvents the vanishing gradient problem. They also question the implications of the architecture changes on the networks learning capabilities.
However, empirical results show that LSTM and its simplified relative Gated Recurrent Unit (GRU) introduced by Cho et al. (2014b) excel in sequence labeling tasks like named entity recognition (Lample et al., 2016), sentiment analysis (Socher et al., 2013), speech recognition (Graves et al., 2013) or POS-tagging (Huang et al., 2015).

3.5 Sequence to Sequence

While gated RNNs are powerful enough to solve tasks where a sequence of variable length is mapped to a fixed amount of labels, they lack the ability to map sequences of variable length to sequences of labels with variable lengths. A solution to this problem was offered by Cho et al. (2014b). They did so by decoupling the input processing from generation, enabling the generation module to produce an arbitrary amount of output symbols. Their proposed architecture consists of two RNNs which is also known as *encoder-decoder network*. The encoder-RNN processes each input symbol yielding an n-dimensional vector, sometimes called thought vector. This vector is then used to initialize the decoder-RNN which in turn generates one symbol per step until it produces a special end-of-sequence symbol. The prediction at each step is conditioned on the state of the decoder and the previous prediction. Sutskever et al. (2014) extend this architecture and propose what is known as the Sequence2Sequence (Seq2Seq) architecture. This architecture has achieved impressive results on several sequence transduction tasks like machine translation (Sutskever et al., 2014; Cho et al., 2014a; Bahdanau et al., 2014), text summarization (See et al., 2017) and constituency parsing (Vinyals et al., 2015).

More recently, Seq2Seq models have been applied successfully in the field of Morphological Reinflection where they secured the first places in the *SIGMORPHON* shared tasks (Cotterell et al., 2016, 2017). Bergmanis and Goldwater (2018) applied the Seq2Seq architecture to lemmatization. They describe their approach as context sensitive, as the encoder processes not only the word form but also 20 characters of left and right context. In contrast to other systems, they do not require morphological- or POS- tags. However, as with Müller et al. (2015), their evaluation is bound to the token level. Schnober et al. (2016) compared pruned CRFs with Seq2Seq architectures and also evaluate on lemmatizing Finnish and German verbs taken from the Wiktionary Dataset (Durrett and DeNero, 2013). Besides limiting the task to verbs, they also lack a qualitative and exhaustive evaluation on the specific task of lemmatization.

3.5.1 Attention

As the standard encoder-decoder architecture compresses its inputs into a fixed size vector, it struggles with long input sequences (Cho et al., 2014a). A way to view this problem is that due to the longer input sequence, the encoder has to compress more information over a longer distance into the same dimensionality. Bahdanau et al. (2014) solve this issue by allowing the decoder to not only access the final state of the encoder but also each intermediate state, which they achieve by using alignment mechanisms.

Vaswani et al. (2017) describe attention as "*mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the*

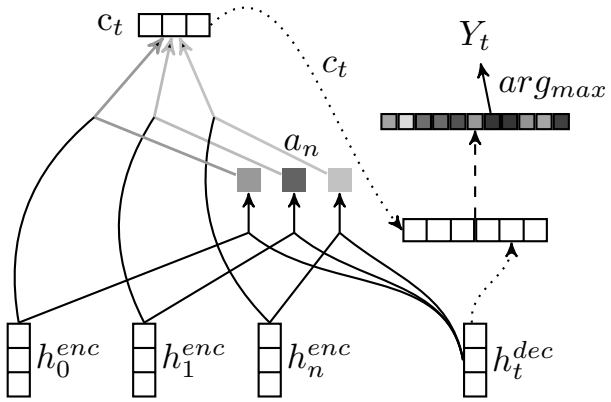


Figure 2: Illustration of Luong attention calculation at decoder step t. Shading denotes values, darker means higher. a_n is the dot product between h_t^{dec} and h_n^{enc}. c_t is the sum of encoder states weighted by a_n. A dotted line describes a concatenation, a dashed line a projection.

weight assigned to each value is computed by a compatibility function of the query with the corresponding key". In the Seq2Seq setup the queries are built from the decoder states, the keys from the encoder states and the values are the encoder states. The main difference between the two commonly used attention variants, Bahdanau et al. (2014) and Luong et al. (2015) lies in the compatibility function. In Bahdanau attention the compatibility score, $a_{i,j}$, between decoder state h_i^{dec} and encoder state h_j^{enc}, is calculated by passing $h_i^{dec} + h_j^{enc}$ through a feed-forward layer. In Luong attention the compatibility function is as simple as the dot product between the two states. Figure 2 is a detailed depiction of Luong attention. As both attentions need to calculate the alignment weights for all encoder states at each decoder step, they have quadratic time complexity in $O(TU)$ where T and U are the lengths of the input and output sequence (Raffel et al., 2017). Raffel et al. (2017) reduce this to linear time with their monotonic Attention. It enforces linear alignments, where the decoder can only move forward in focusing on encoder states Figure 3 illustrates alignment and the attentional weights when translating an English sentence to French.

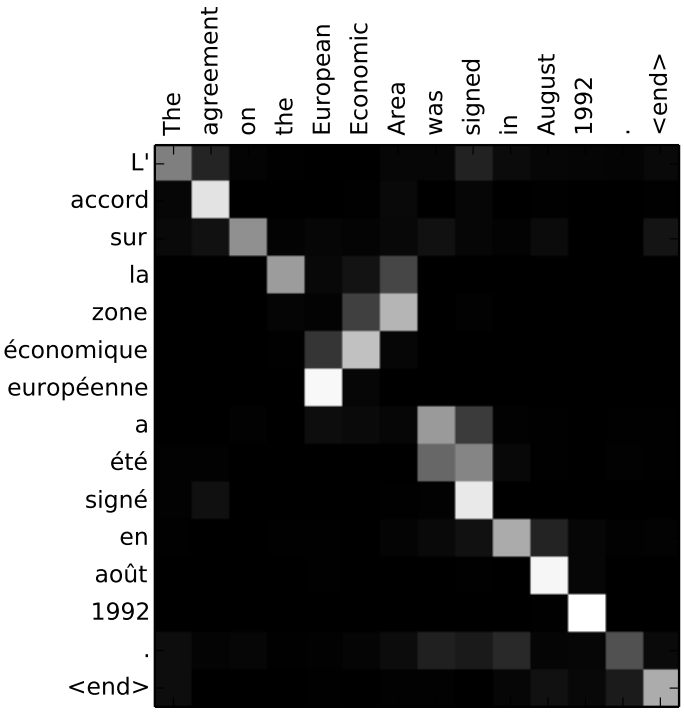


Figure 3: Illustration of attention weights when translating an English sentence to French. Brighter spots indicate a higher attention weight. Figure taken from: Bahdanau et al. (2014)

4 Models

In this section, we will describe the five proposed Seq2Seq variants: (i) *Ohnomore-Base*, (ii) *Ohnomore-CRF*, (iii) *Ohnomore-Att*, (iv) *Ohnomore-Att-CRF* and (v) *Ohnomore-Att-Morph*. They all have been realized in Tensorflow (Abadi et al., 2016) using the Seq2Seq framework. Another common property is a character-level LSTM both in encoder and decoder, motivated by the need to transfer morphological information gathered in the encoder to the decoder, constituting a long distance dependency. In (8), for example, the information of the separable prefix "wieder" and of the inflectional "t" in the suffix need to be available for the decoder to drop the separable prefix "wieder" or respectively replace the inflectional suffix "t" by "n".

(8) wiedereröffnet → eröffnen
reopened *reopen*

Ohnomore-Base closely resembles Sutskever et al. (2014). It deviates from it as we do not reverse the input sequence and replace the single start token in the decoder with the part-of-speech of the input form, which enables the model to disambiguate some homographs.

Ohnomore-CRF aims to correct improbable character sequences in the decoder by combining it with a linear-chain Conditional Random Field.
Ohnomore-Att extends **Ohnomore-Base** with a Luong-style monotonic Attention mechanism (Luong et al., 2015; Raffel et al., 2017), exploiting the mostly linear alignment between forms and lemmas.
Ohnomore-Att-CRF combines **Ohnomore-CRF** and **Ohnomore-Att** by utilizing both attention and a linear-chain CRF.
Ohnomore-Att-Morph differs from **Ohnomore-Att** by adding morphological tags to the input, providing additional disambiguating material. The embedded morphological and part-of-speech tags are concatenated with the final state of the encoder, resulting in an $d + p + (m * n)$ dimensional vector, where d is the state size of the encoder, p and m the embedding size of parts-of-speech and morphological tags and n the maximal number of morphological tags encountered during training. This vector is then fed through a feedforward layer with the SELU activation function (Klambauer et al., 2017), resulting in a vector with the dimensionality of the decoder's state size, which is the initial state of the decoder. As the POS are provided this way, we use a single start token in the decoder for all POS. Figure 4 is a detailed illustration of **Ohnomore-Att-Morph**.

We also experimented with stacked RNNs, beamsearch (Wu et al., 2016), a bi-directional encoder as described in Bahdanau et al. (2014), Bahdanau-attention (Bahdanau et al., 2014) and non-monotonic attentions, however, preliminary results showed that the uni-directional encoder and monotonic Luong attention lead to better results while being less complex.

5 Evaluation and Data

In contrast to other recent work in lemmatization like Müller et al. (2015) or Bergmanis and Goldwater (2018) we decided to evaluate on types instead of tokens. We did so because we suspect that token-based evaluation is biased towards getting frequent tokens right. Moreover it is to be expected that a token which ends up both in the training and validation set will predicted right, simplifying the task.

We evaluated on two German datasets: TüBa-D/Z (Telljohann et al., 2004) which contains newspaper articles of the Tageszeitung (TAZ)[1] and as out-of-domain test on NoSta-D (Dipper et al., 2013) a corpus containing non-standard variations of German. Further, we evaluated on Dutch using the Lassy-small treebank (Van Noord et al., 2006, 2013).

As a strong baseline we trained two variants of Lemming (Müller et al., 2015) (*Lemming-Base* and *Lemming-List*). *Lemming-Base* only utilizes its built-in features, including several alignment, edit-tree and lexical features. As Lemming supports the addition of arbitrary features we also trained *Lemming-List* which adds a word list.

Since Lemming's and *Ohnomore-Att-Morph's* performance heavily depends

[1] http://taz.de/

on morphological tags and the non-morph models need to learn word internal morphology, we did not want to train on gold-morphology tags. We then tagged TüBa-D/Z and Lassy-small using 5-fold jackknifing. Jackknifing is a method commonly used in NLP, inspired by the statistical technique with the same name presented in Quenouille (1956) and Tukey (1958). Jackknifing trains on $n-1$ folds while leaving one out to be annotated. After repeating this procedure n times a fully machine-tagged corpus has been obtained. Besides tagging, we filter out duplicate tokens such that each combination of form, lemma, POS and morphological-tags is unique. We decided to use the non-gold morphological tags while removing the duplicates to evaluate both morph and non-morph models on the same data sets. Further, we filtered out closed class words and irregular inflected forms, since we consider it as impossible to infer the lemma when using on a type level disjoint sets. Additionally, we suspect that both irregular forms and closed class words can be easily lemmatized using dictionaries.

After preparing the data in this way, we perform 10-fold cross-validation. All used hyperparameters are reported in Appendix A.

5.1 Dutch

Our Dutch dataset is the Lassy-small treebank. It contains 1 million tokens with syntactical and morphological annotations. After removing duplicates, closed class words and irregular verbs we arrive at 98441 types. Our list of irregular verbs is extracted from the regularity dataset (Tabak et al., 2005; Baayen and Prado Martin, 2005) of the languageR package[2] and Wiktionary[3], resulting in 336 irregular verbs. The closed class tags are included in Appendix B.1. We extracted the word list for *Lemming-List* from the Lassy-big treebank.

5.2 German

Our German training data is TüBa-D/Z. It contains 1.8 million tokens with syntactical and morphological information, removing duplicates, closed class words and irregular verbs yielded 219707 types. We removed a list of 2039 irregular verbs extracted from Celex German (Baayen et al., 1993). The closed class tags are included in Appendix B.2. The word list for *Lemming-List* was taken from `https://sourceforge.net/projects/germandict/files/`[4].

5.2.1 NoSta-D

To test out-of-domain performance we used the NoSta-D corpus. It contains 33.278 annotated tokens. After removing duplicates, closed class words and irregular verbs we arrived at 5057 types.

[2] available at `https://rdrr.io/cran/languageR/`

[3] `https://en.wiktionary.org/wiki/Category:Dutch_irregular_verbs`

[4] accessed on 24.7.2018

Ohnomore-Att-Morph

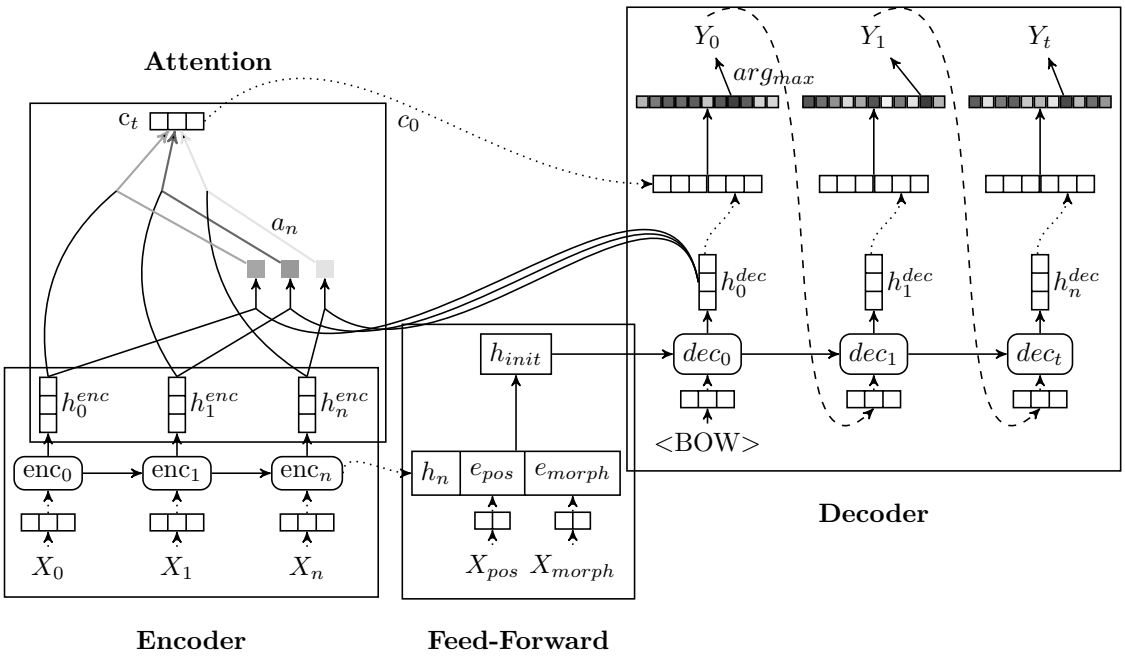


Figure 4: Illustration of *Ohnomore-Att-Morph*. $X_0..X_n$ denote input symbols, $Y_0..Y_n$ predictions. Rectangles with rounded edges denote RNN states. Rectangles with cells are vectors. A single rectangle denotes a scalar. The connection of a input or prediction symbol with a vector denote an embedding. Dotted lines express the concatenation of vectors. Shading expresses values, darker means higher. The shaded line meeting a solid line denotes a scalar multiplication of the vector. Two lines from two vectors meeting denote the dot product. A solid line connecting two vectors of differing size denotes a projection. a_n denotes the attentional weight on the encoder output n at decoder step t.
c_t denotes the context vector, the result of the attention calculation at decoder step t.
For reasons of readability, we omitted the attention calculation at dec_1 and dec_t.

6 Results and Discussion

	TüBa-D/Z	**NoSta-D**	**Lassy**
Oh-Base	93.16%	77.79%	91.34%
Oh-Att	96.16%	78.82%	94.07%
Oh-Att-Morph	**96.92**%	**79.73**%	**96.25**%
Oh-CRF	70%	65.18%	
Oh-CRF-Att	89%	76.33%	
Lemming-Base	96.8%	80.62%	95.6%
Lemming-List	**97.01**%	**81.39**%	**95.97**%

Table 2: Results of the evaluation on TüBa-D/Z, NoStaD and Lassy-Klein. In bold are the best results of the Lemming / Ohnomore models. Bold and underlined are the best overall results. Underlined without bold the best non-morphology model.

In Table 2 we observe that all attention Seq2Seq models outperform the non-attention models. Furthermore we see that there are serious issues with applying a CRF-layer on the output of the decoder as it results in non-competitive performance. Besides these Seq2Seq internal differences, we also find that the best performing models across all datasets are the ones that take morphological information into account, with *Lemming-Base* yielding better results than *Oh-Att* across all sets. Overall, we find that *Lemming-List* and *Oh-Att-Morph* show similar performance, with *Lemming-List* being the stronger choice for both German sets, while *Oh-Att-Morph* performs slightly better for Dutch. The difference in performance on Dutch might be influenced by an incomplete list of the irregular verbs.

Figure 5 illustrates the learning curves of *Oh-Att-Morph* and *Lemming-List* on TüBa-D/Z when provided with incrementally more training data. It should be noted that we validated on the non-training split, hence with more used training data, the validation set shrinked. As both curves start to level off at 90%, it seems that both *Lemming-List* and *Oh-Att-Morph* become saturated with the available data.

In Figure 6 we show how the attention mechanism achieves a diagonal alignment between the input and the lemma. It can also be seen that lookaheads happen at the usual position of inflections, namely the beginning of the word, where a separable prefix is removed and then at the end of the word the common location of finite verb inflections.

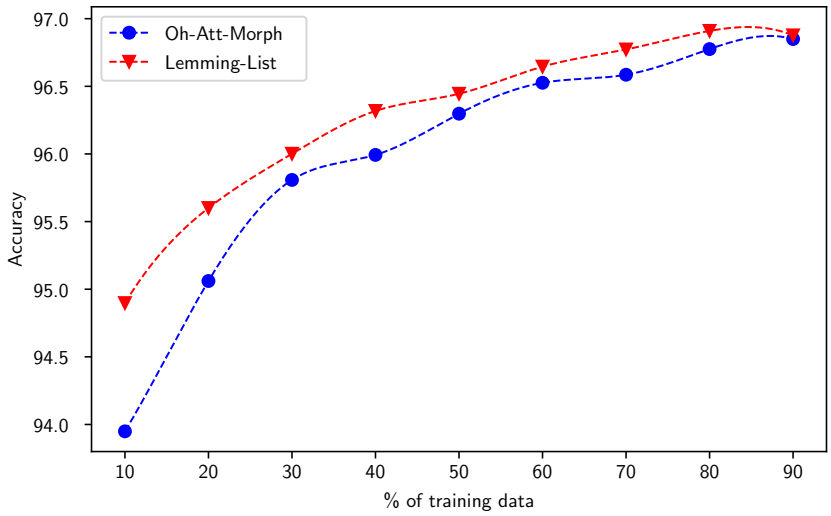


Figure 5: Learning curves of *Oh-Att-Morph* and *Lemming-List* when provided with incrementally more training data.

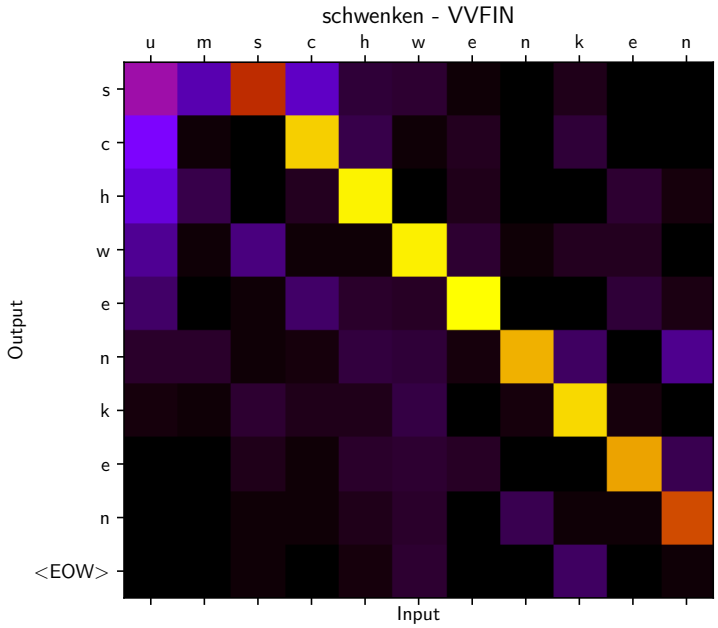
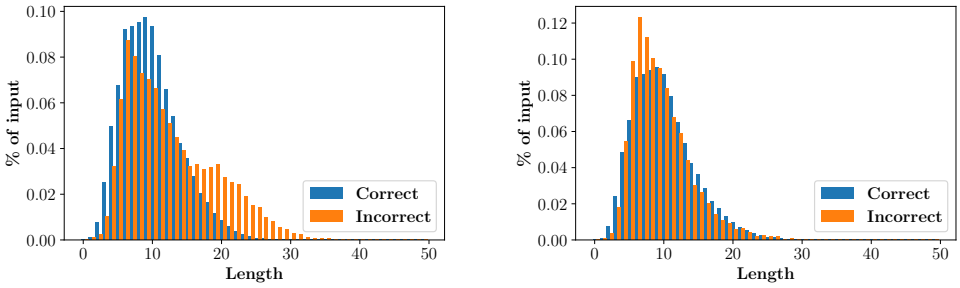


Figure 6: Illustration of the alignment and attentional weights between the separable verb "umschwenken" and the predicted lemma "schwenken". The input is on the y-axis, the output on the x-axis. Brighter spots indicate a higher attentional weight. The expected lemma and the POS are in the title.

6.1 Sequence Lengths

Apart from the effect on the accuracy, we also observe that the attentional models are able to deal with longer input. When comparing Figure 6-a and Figure 6-b, we clearly see how the distribution of input lengths that lead to false predictions is skewed towards longer lengths for *Oh-Base* while with *Oh-Att* no such effect can be seen. This again confirms the finding of Cho et al. (2014a), that an encoder without attention struggles to compress longer sequences in a fixed size representation.



(a) Oh-Base (b) Oh-Att

Figure 7: Histograms showing that Oh-Att deals better with longer input sequences than Oh-Base. On the y-axis percentage of inputs. On the x-axis input lengths. Blue are the inputs that lead to a correct prediction, orange those which lead to an incorrect one.

6.2 Morphology

The margin between the morphology and non-morphology model is partially explained by ambiguous form-lemma-POS triples. In TüBa-D/Z, there are 635 form-pos pairs associated with more than one lemma, resulting in a total of 1311 ambiguous cases. Lassy-small contains 797 form-POS pairs with multiple lemmas and a total of 1616 ambiguous examples.

Table 3 reports the error rates of *Oh-Att*, *Oh-Att-Morph* and *Lemming-List* on the ambiguous cases, across all folds. It can be seen that *Oh-Att-Morph* clearly outperforms *Oh-Att*. Surprisingly *Lemming-List*, also taking explicit morphological information into account, scores as the worst of the three.

	TüBa-D/Z	**Lassy**
Oh-Att	60.87%	63.61%
Oh-Att-Morph	52.94%	52.14%
Lemming-List	75.80%	80.92%

Table 3: Error rates on form-pos pairs with multiple lemmas.

After investigating how the ambiguous cases were split among the folds, i.e. whether several cases landed in the same fold or if one was part of the training set, we find that *Lemming-List* rather sticks to a seen lemma than to generate an unseen one. We discover quite some cases in which *Lemming-List* disregards disambiguating morphological information and outputs a known lemma that is clearly contradicted by the morphology. *Oh-Att-Morph*, in contrast, relies more on morphological information and does not exhibit these problems, however, due to its dependence on the tags, it propagates errors created during the tagging process. The morphologically uninformed *Oh-Att* does not stick to known lemmas as well and outputs the same lemma more consistently across the folds, regardless of what was seen during training.

6.3 CRF

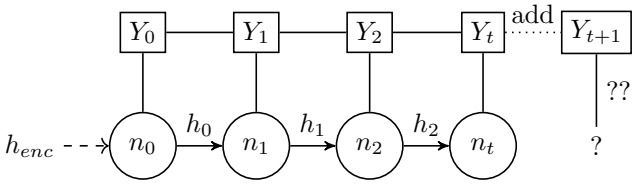


Figure 8: Illustration of the CRF's limitations with regard to additions. $n_{0..t}$ represent the observations (decoder steps), $h_{0..2}$ the hidden states of the decoder-RNN. $Y_{0..n}$ the label sequence. The add arc demonstrates how out-voting the end-of-word symbol leads to a label Y_{t+1} without a corresponding observation n_{t+1}.

The poor performance of the CRF models indicates serious limitations with the the CRF decoder. After further investigation, we realized, that the decoder-RNN already determines most of the lemma. Different than in other CRF applications where we have a fixed number of inputs and corresponding labels, the inputs here are the product of the decoder-RNN, where the predicted score distribution over characters at a certain timestep is conditioned on the previous output and the hidden state. As such they are subject to erroneous productions, including additional or missing characters.

Suppose the output of the decoder-RNN is a series of $n_0, n_1 ... n_t$ predictions, where each n is a vector of real numbers containing the unnormalized score distribution over the output vocabulary and t is the timestep where the decoder assigned the highest score to the end-of-word symbol. What becomes clear now is, that we only got access to conditional classifications up to timestep t. Should the decoder-RNN mistakenly cut off a suffix, assigning only a slightly higher score to the end-of-word symbol, which the transition probabilities could outvote, the CRF would run out of observations as there are no further timesteps following t for which the RNN produced score distributions.Figure 8 demonstrates this limitation, as there is no corresponding observation to the label Y_{t+1}.

In (9) the error is not a missing suffix, but an additional character in the middle of the word.

(9) gelaufen → *lauffen (laufen)
walked **walk*

The goal here would be to delete the character, while keeping the rest of the sequence. As a CRF computes the highest scoring sequence of labels over a set of observations, and, in this case, the observations are a chain, the only way to delete a character would be to skip an observation as shown in Figure 9. Inserting a character in the middle of a lemma combines the two aforementioned limitations, as it would mean to add some prediction, effectively shifting one of the following predictions into a place where no RNN output is available.

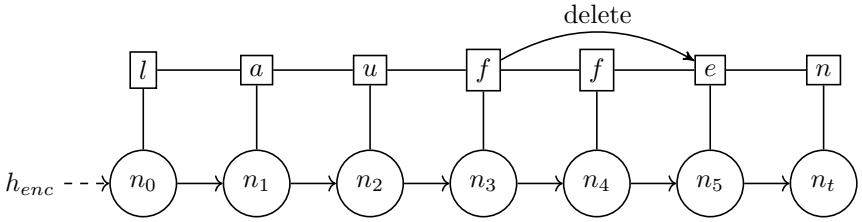


Figure 9: Illustration of the CRF's limitations with regard to deletions. $n_{0..t}$ represent the observations (decoder steps), $h_{0..2}$ the hidden states of the decoder-RNN. $Y_{0..n}$ the label sequence. The delete arc shows that if the superfluous character should be deleted, it is necessary to skip observation n_4.

6.4 Error Analysis

Model	Correct	Unsolv-Misc	Trunc	Colloq	Cor	Ambig	NE	Sep	Solv-Misc	-en
Oh-Att	96.66%	0.04%	0.05%	0.13%	0.35%	0.57%	0.61%	0.13%	0.98%	0.53%
Oh-Att-Morph	97.21%	0.04%	0.05%	0.13%	0.31%	0.39%	0.52%	0.10%	0.76%	0.47%
Lemming-List	97.47%	0.06%	0.05%	0.07%	0.12%	0.51%	0.42%	0.30%	0.65%	0.32%

Table 4: The result of the analysis of 20.029 sampled predictions from *Oh-Att*, *Oh-Att-Morph* and *Lemming-List*. Full explanation of error classes with examples in Appendix C.

For a better quantification and comparison of the best scoring models we sampled 20.029 examples from TüBa-D/Z and classified the incorrect portion of the predictions of *Oh-Att*, *Oh-Att-Morph* and *Lemming-List* for these into 9 classes. False predictions which are identical across all models will be discussed in the next section.

Besides these 9 fine grained classes we also grouped the errors by whether we consider them solvable or not. Within the unsolvable group are cases with

typos in the target-lemma, colloquial- and dialect- forms with canonicalized lemma and truncated forms with full lemma. As borderline solvable we consider cases where a typo in the form has been corrected in the lemma, named entities, separable verbs and several instances of ambiguity, like the plural noun phrase syncretism or homographs. We decided to divide the solvables into two classes, one general, and another one to measure how big the impact of the rather ambiguous suffix "*-en*" is on the different models. A full explanation of the error classes with examples can be found in Appendix C.

The results of our analysis are presented in Table 4. We see that *Lemming-List* performs better with correcting errors and lemmatizing named entities. We suspect that Lemming's word list plays a role in both cases. With malformed input it indicates whether its potential output is wellformed. With named entities on the other hand it helps to decide if a inflection in the suffix is part of the name, like in "Sotherby's". Further, we find that *Lemming-List* recognizes the inflectional parts of the suffix "*-en*" better than the Seq2Seq models. For morphological ambiguities we confirm the finding of 6.2 and see that *Oh-Att-Morph* is better at incorporating disambiguating morphological information. Interestingly *Lemming-List* performs worse than both Sequence2Sequence models when dealing with separable verbs. A possible explanation is that Lemming only creates edit-trees for operations that have been seen often enough. Due to training on types it is possible that for these rather rare forms no edit-trees were produced.

6.4.1 Intersection of Errors

Unsolv-Misc	**Trunc**	**Colloq**	**Cor**	**Ambig**	**NE**	**Sep**	**Solv-Misc**	**-en**
6.0%	0.0%	7.6%	47.8%	4.6%	15.5%	2.3%	12.6%	6.7%

Table 5: The result of the analysis of 341 sampled errors from the intersection of identical errors produced by the *Oh-Att*, *Oh-Att-Morph* and *Lemming-List*. Full explanation of error classes with examples in Appendix C.

To further examine the errors, we built the intersection of errors where all models produced the same false prediction, this accounted for 3179 errors. We then sampled 341 cases from this intersection and analyzed them. The results are presented in Table 5. We find that 47.80% (163) of these were due to spelling errors within the form that have been corrected. As can be seen in the first and last example of Table 6, both models correctly cut the inflectional suffix "*-en*", but fail to correct the spelling mistakes. We suspect that the misspellings are not frequent enough (a) to reach Lemming's threshold to introduce a new edit-tree and (b) for the Seq2Seq models to pick up the signal, giving incentive to change non-inflectional parts of the words.

Form	Lemma	Prediction	POS
humantären	humanitär	humantär	ADJA
allerding	allerdings	allerding	ADV
sozialderzernenten	sozialdezernent	sozialderzernent	NN

Table 6: Spelling errors in forms in TüBa-D/Z that have been corrected in the lemma with prediction of both *Lemming-List* and *Oh-Att-Morph*.

7 Conclusion

In this thesis, we have proposed the application of several variations of the Seq2Seq architecture for lemmatization. We evaluated their effectiveness on German and Dutch and also test out-of-domain for German. The results show that the Seq2Seq architectures achieve competitive performance. More precisely we have found that (i) the attentional neural models outperform the non-attentional ones, (ii) the CRF-decoder shows the worst performance of all models, (iii) all models that explicitly incorporate morphological information outperform the non-morphological models, (iv) *Lemming-List* has an advantage if the input is not wellformed and (v) *Oh-Att-Morph* uses the explicit morphological information to disambiguate between syncretic forms, while *Lemming-List* puts a higher weight on whether a lemma is known.

Moreover, we theorize that (i) *Lemming-List's* word list informs the classifier whether a candidate is wellformed or not, while the Seq2Seq models lack this ability and (ii) applying a CRF layer to the decoder is not viable, as operations like additions and deletions are not possible since the number of observations is fixed as the decoder stops.

For future work we want to tackle the problem with malformed input, as we saw that almost 50% of the errors shared by *Ohnomore-Att*, *Ohnomore-Att-Morph* and *Lemming-List* were due to these cases and we expect that making the models more robust towards malformed input will benefit their potential when being used to lemmatize noisy web corpora. Possible approaches include incorporating a word list or more global optimization algorithms like Minimum Risk Training (Shen et al., 2015) or MIXER (Ranzato et al., 2015). We also suspect that improving on the morphological tagging will yield better results. Here it could be worthwhile to explore whether assigning morphological tags and lemmas jointly yields the same improvements as Müller et al. (2015) and Chrupała et al. (2008) report.

While we have shown that the morphologically informed Seq2Seq variant achieves competetive performance, we also plan to explore possibly simpler neural architectures as classifiers for edit scripts following Chrupała (2006) and Müller et al. (2015).

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*. volume 16, pages 265–283.

R Harald Baayen, Richard Piepenbrock, and Rijn van H. 1993. The CELEX lexical data base on CD-ROM .

R Harald Baayen and Fermin Moscoso del Prado Martin. 2005. Semantic density and past-tense formation in three germanic languages. *Language* 81(3):666–698.

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

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

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)*. volume 1, pages 1391–1400.

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

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Grzegorz Chrupała. 2006. Simple data-driven context-sensitive lemmatization. *Procesamiento del lenguaje natural, nº 37 (sept. 2006), pp. 121-127* .

Grzegorz Chrupała, Georgiana Dinu, and Josef Van Genabith. 2008. Learning morphology with morfette .

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, et al. 2017. Conll-sigmorphon 2017 shared task: universal morphological reinflection in 52 languages. *arXiv preprint arXiv:1706.09031* .

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. 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.

Stefanie Dipper, Anke Lüdeling, and Marc Reznicek. 2013. Nosta-d: A corpus of german non-standard varieties. *Non-Standard Data Sources in Corpus-Based Research* (5):69–76.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 484–490. http://aclweb.org/anthology/P18-2077.

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.

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

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, pages 6645–6649.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR* abs/1508.01991. http://arxiv.org/abs/1508.01991.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. *CoRR* abs/1706.02515. http://arxiv.org/abs/1706.02515.

Wolfgang Kruase and Gerd Willée. 1981. Lemmatizing german newspaper texts with the aid of an algorithm. *Computers and the Humanities* 15(2):101–113.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data .

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* .

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Guido Minnen, John Caroll, and Darren Pearce. 2001. Applied morphological processing of english. *Natural Language Engineering* 7(3):207–223. https://doi.org/10.1017/S1351324901002728.

Thomas Müller. 2015. *General methods for fine-grained morphological and syntactic disambiguation*. Ph.D. thesis, Ludwig-Maximilians-Universität München. http://nbn-resolving.de/urn:nbn:de:bvb:19-182637.

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.

Thomas Müller, 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.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR, abs/1211.5063* .

Martin F Porter. 1980. An algorithm for suffix stripping. *Program* 14(3):130–137.

Martin F Porter. 2001. Snowball: A language for stemming algorithms. `http://snowball.tartarus.org/texts/introduction.html`.

Maurice H. Quenouille. 1956. Notes on bias in estimation. *Biometrika* 43(3/4):353–360. http://www.jstor.org/stable/2332914.

Colin Raffel, Minh-Thang Luong, Peter J Liu, Ron J Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. *arXiv preprint arXiv:1704.00784* .

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* .

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. Smor: A german computational morphology covering derivation, composition and inflection. In *LREC*. Lisbon, pages 1–263.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. *arXiv preprint arXiv:1610.07796* .

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368* .

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*. volume 1, pages 83–91.

Rico Sennrich and Beat Kunz. 2014. Zmorge: A german morphological lexicon extracted from wiktionary. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), Reykjavik, Iceland.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433* .

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. pages 1631–1642.

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

Charles Sutton and Andrew McCallum. 2010. An introduction to conditional random fields. *stat* 1050:23.

Wieke Tabak, Robert Schreuder, and R Harald Baayen. 2005. Lexical statistics and lexical processing: semantic density, information complexity, sex, and irregularity in dutch. *Linguistic evidence—Empirical, theoretical, and computational perspectives* pages 529–555.

Heike Telljohann, Erhard Hinrichs, Sandra Kübler, and Ra Kübler. 2004. The tüba-d/z treebank: Annotating german with a context-free backbone. In *In*

Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004). Citeseer.

John W. Tukey. 1958. Abstracts of papers. *Ann. Math. Statist.* 29(2):614–623. https://doi.org/10.1214/aoms/1177706647.

Gertjan Van Noord, Gosse Bouma, Frank Van Eynde, Daniel De Kok, Jelmer Van der Linde, Ineke Schuurman, Erik Tjong Kim Sang, and Vincent Vandeghinste. 2013. Large scale syntactic annotation of written dutch: Lassy. In *Essential Speech and Language Technology for Dutch*, Springer, pages 147–164.

Gertjan Van Noord, Ineke Schuurman, and Vincent Vandeghinste. 2006. Syntactic annotation of large corpora in stevin. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC), Genoa, Italy*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. pages 5998–6008.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2773–2781.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE* 78(10):1550–1560.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144. http://arxiv.org/abs/1609.08144.

A Hyperparameters

A.1 Seq2Seq

Optimizer: Adam (Kingma and Ba, 2014) with learning rate 0.03

Early stopping after: 10 epochs

A.1.1 Non-morph

Character-embedding size: 32

LSTM-size: 192

Input-dropout: 0.8

LSTM-dropout: 0.9

A.1.2 Morph

Character-embedding size: 100

POS-embedding size: 50

Morph-embedding size: 30

LSTM-size: 300

Input-dropout: 0.9

LSTM-dropout: 0.5

A.2 Lemming

Classifier: Perceptron

Iterations: 10

B Closed Class Tags

B.1 Dutch

det	pron	punct
prep	tag	pp

B.2 German

- -	KOUI	PRELS	PWS
$(	KOUS	PRF	VAFIN
$,	PDAT	PROAV	
$.	PDS0	PROP	VAIMP
APPO	PIAT	PTKA	VAINF
APPR	PIDAT	PTKANT	
APPRART	PIS	PTKNEG	VAPP
APZR	PPER	PTKVZ	VMFIN
ART	PPOSAT	PTKZU	
KOKOM	PPOSS	PWAT	VMINF
KON	PRELAT	PWAV	VMPP

C Error Classes

Error Class	Explanation	Form	Lemma	POS	Prediction	Model
Unsolv-Misc	unsolvable errors, spelling errors in target lemma etc.	enthüllungen	enthüllunge	NN	enthüllung	*Lemming-List*
Cor	typo in form, corrected in lemma	profannen	profanes	NN	profanne	*Oh-Att-Morph*
Colloq	colloquial language or dialect normalized	nia	nie_	ADV	nier_	*Lemming-List*
Trunc	truncated form with full lemma	-tätern	ns-täter	NN	statertäter	*Oh-Att-Morph*
Ambig	nouns that can be read both as nominalized verbs and proper noun, ambiguous gender, etc.	trommeln	trommel	NN	trommeln	*Oh-Att-Morph*
NE	named entities	watson's	watson	NE	watson's	*Oh-Att*
Sep	separable verbs	durchhört	hören	VVFIN	durchhören	*Oh-Att*
Solv-Misc	general solvable errors	kerle	kerl	NN	kerle	*Oh-Att*
-en	errors related to the suffix "*-en*"	achselzucken	achselzucken	NN	achselzuck	*Oh-Att-Morph*